Pattern-Theoretic Foundations of Automatic Target Recognition in Clutter AFOSR grant F49620-03-1-0340 Final Report

Principal Investigator: Aaron Lanterman
School of Electrical and Computer Engineering
Georgia Institute of Technology, Mail Code 0250
Atlanta, GA 30332
(314)-385-2548, lanterma@ece.gatech.edu

May 24, 2007

Abstract

This effort advanced the art of applying Grenander's pattern theory to automatic target recognition (ATR) problems. We extended jump-diffusion ATR algorithms to accommodate unknown infrared camera calibration effects and include more stable diffusion procedures for pose refinement, and developed flexible shape models to accommodate clutter. We also developed performance bounds on estimation and recognition performance for low-frequency radar data, single-image laser radar data, and "point cloud" 3-D data assembled from multiple sources. Further work explored data fusion using the "probability hypothesis density" approach.

Contents

1 Objectives

This research project seeks novel applications of Grenander's pattern theory to problems of Automatic Target Recognition (ATR) in clutter. Model-based algorithms can be powerful, but they can also be fragile if there is a substantial mismatch between the reality and the model. Previous applications of pattern theory to ATR for infrared and laser radar data have considered scenes consisting of targets from a known library against a simple background. While this may be sufficient for relatively simple scenarios, such as tanks against a desert background, such simple "target/background" parameterizations will have difficulty with more cluttered scenes. One fundamental aspect of studying such systems is development of fundamental lower bounds on their performance; such bounds may be used to optimize system parameters.

1.1 Philosophy

Our pattern-theoretic approach to ATR might be thought of as "recognition through simulation" or "recognition through synthesis." In traditional ATR development, simulation plays an important role in the creating numerous data sets to test traditional algorithms. In contrast, in our Grenander-inspired approach, simulation plays a role inside the ATR algorithm itself.

In the field, an ATR system collects some data about an underlying scene of interest. Ideally, we would have a Magic Sensor that says "there is a T62 at this latitude and longitude, and an M60 at this other latitude and longitude." Our real sensors must collect data subject the vagaries of nature and the sensor. In the case of laser radar, we see at targets through the effects of obscuration and perspective projection. The sensor

adds additional complications such as sampling effects, range noise, and anomalous ("garbage") pixels.

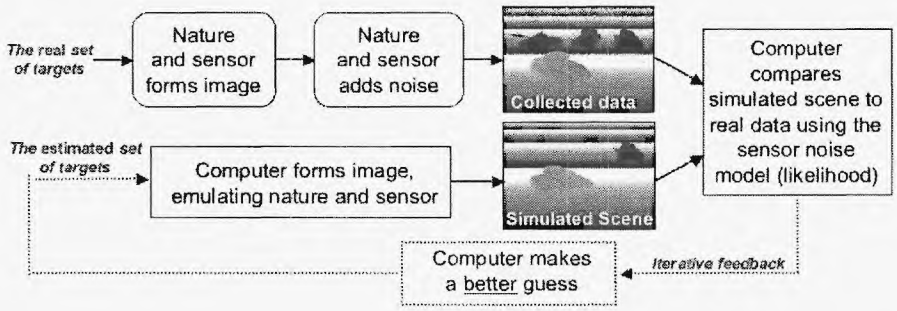


Figure 1: High-level view of our approach to designing ATR algorithms.

As illustrated in Figure 1, our approach to ATR involves simulating a scene that would be generated by a hypothesized set of targets (including types, positions, orientations, etc.), but not including noise. The effect of noise on the real data is encapsulated in the "sensor likelihood" function, which compares our hypothetical computer-generated image with the real data. If the likelihood is high, then the computer-generated image and the real data are a close match, and our hypothesis was a good one. However, our initial hypothesis is probably not that good, and our hypothesized scene does not match the real data well; hence, we invoke a feedback loop, by which the hypothesis is refined, and new hypothesized target sets are formed. The process repeats, with the algorithm continually trying new target types, refining their positions and orientations, etc., trying to get its simulated scene to most closely match the real data.

Note that this process just involves simulating scenes and comparing them to the real data with a sensor likelihood. There are no separate stages of denoising, edge detection, feature extraction, etc. as often found in traditional ATR algorithms.

Although Figure 1 uses laser radar as an example, this philosophy may be used on any sort of sensor, as long as a model for generating data with that sensor is available. Having accurate sensor models, particularly for tasks like scene generation, is important. In this approach, it is vital that we get to know our sensors.

1.2 Context of Effort

Work on this Pattern Theoretic ATR (PTATR) effort under AFOSR funding began in mid-August, 2003. Although the initial effort primarily focused on ATR with infrared data, it split into four related threads. The first two, ATR with infrared data (Section 2.1) and ATR with 3-D data (Section 2.2), constitute the bulk of the PTATR effort. The second two tasks, the creation of performance bounds (Section 2.3) and data fusion using finite-set statistics (Section 2.4), constitute synergistic adjuncts with other efforts. Additional effort was expended towards developing sensor simulation software for use

by the general ATR community, as described in Section 5.2. A subcontract from Jacobs Sverdrup allowed Jason Dixon to spend some time working directly with personnel at Wright-Patterson Air Force Base during the development of the LADAR simulator.

To best leverage the government's support and give AFOSR maximal impact for its investment, work in this area has continued past the September 2006 end date via discretionary funds provided to Aaron Lanterman through the Demetrius T. Paris Junior Professorship.

2 Accomplishments

2.1 ATR with Infrared Data

As described in Appendix F, we are exploring different ways of extending previous models by supplementing the 3-D CAD models of specific targets with flexible models that can accommodate clutter objects not found in the algorithm's target library. Some emphasis has been placed on moving away from strict diffusion implementations, since diffusions involve difficult choices involving step sizes (both in terms of discretizing the diffusion itself and in terms of numerical computations of derivatives); as described in Section 2.1.1, we have explored achieving the effect of diffusions via "little jumps" that avoid some of these complications.

In jump-diffusion algorithms for Bayesian inference, the jumps are typically responsible for handling large changes in the hypothesized configuration, such as the addition or deletion of a target or a change of target type, while the diffusions refine continuous parameters, such as position and orientation. Various kinds of jumps are illustrated in Figure 2.

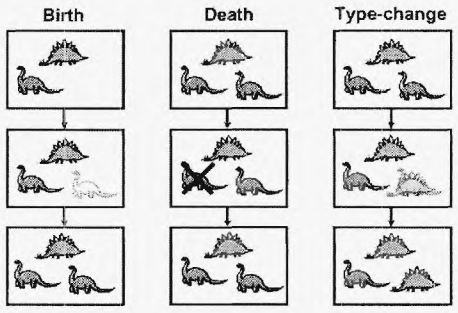


Figure 2: Illustration of jump moves to add hypothesized targets, remove hypothesized targets, and change hypothesized target types.

Our previous work exploring jump-diffusion algorithms for automatic target recognition from infrared scenes assumed that the radiant characteristics of the objects were

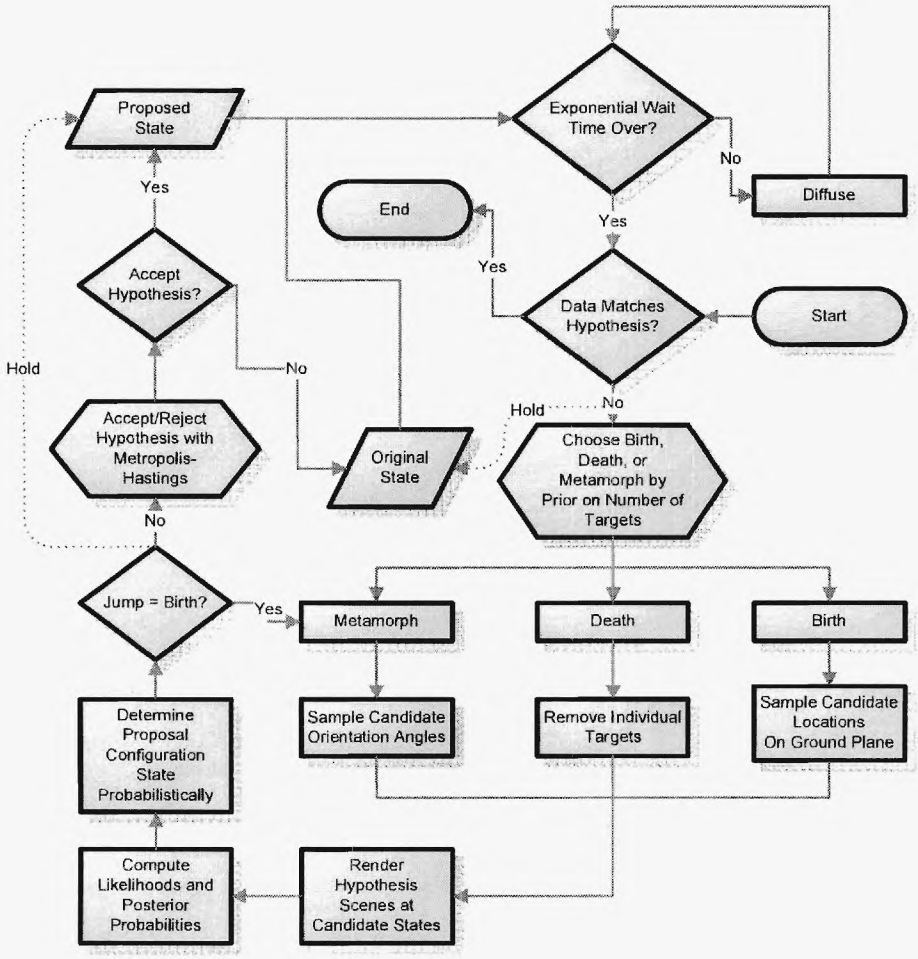


Figure 3: Flowchart of legacy jump-diffusion algorithm for pattern-theoretic ATR with infrared data.

known. The radiant characteristics of real objects varies widely depending on environmental conditions and operational history; for instance, whether the engine has been running and how hard, whether it is a cloudy or sunny day, whether the gun has just been fired, etc. To accommodate this thermal variability within the Grenander framework, we have incorporated eigentank models into the jump-diffusion code. The eigentanks are defined over the full 3-D surface of the target, although only a portion of that surface is projected onto the detector. The eigentank models are derived from a principal component analysis of thermal profiles simulated under a wide range of conditions, and characterize how different parts of a target "heat up" or "cool down" together. Figure 3 gives a flowchart of our jump-diffusion algorithm for ATR with infrared data, as it roughly stood before the beginning of our PTATR effort. Figure 4 shows where the new advances developed under this effort, including modifications

described in Appendix B and Sections 2.1.1 and 2.1.2, fit into the overall algorithm.

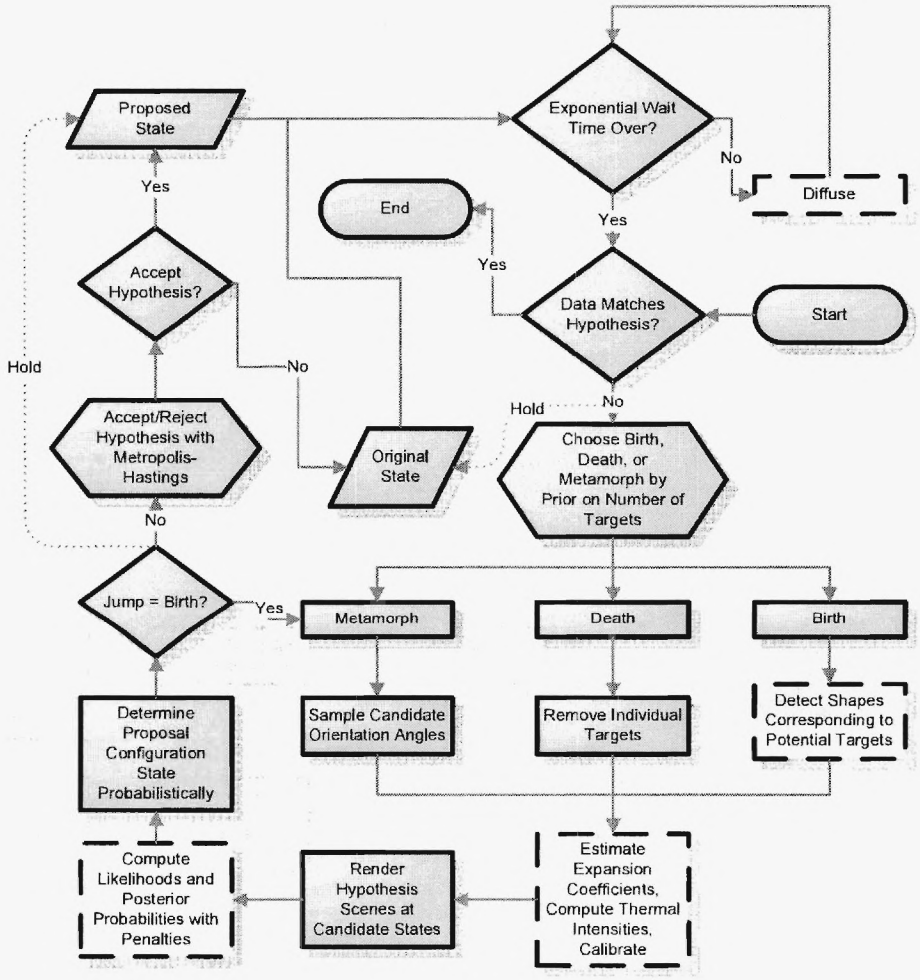


Figure 4: Flowchart illustrating enhancements to the pattern theoretic ATR with infrared data developed under the current effort.

2.1.1 Improved Diffusion Approaches

Background on previous Langevin-based diffusion algorithms: The pattern-theoretic ATR algorithm employed in our studies uses a type of jump-diffusion process[1] to iteratively estimate target characteristics and pose present in a given image. The pose parameters are used to render hypothesized scenes through a set of OpenGL routines, which are compared to the original data image using a likelihood function based on forward-looking infrared (FLIR) or laser radar (LADAR) sensor statistics. The jumps determine the size of the parameter space, i.e., the number of targets, while diffusions update pose parameter estimates.

During the intervals between jumps, the diffusion process takes over and adjusts the continuous configuration pose parameters ($[x, y, \theta]$ – two ground-based coordinate positions and an orientation angle) by small amounts to better align the hypothesized targets with the corresponding detected targets in the data. Diffusions are accomplished using the Langevin stochastic differential equation (SDE):

$$d\mathbf{C}_N(\tau) = \nabla_{\mathbf{C}_N} H[\mathbf{C}_N(\tau)|\mathbf{d}]d\tau + \sqrt{2}dW_N, \tag{1}$$

where W_N is a Wiener process and $H[\mathbf{C}_N(\tau)|\mathbf{d}]$ is the logposterior for data $\mathbf{d}$ associated with the configuration parameter vector $\mathbf{C}_N$, which contains the configuration parameters for N targets with fixed classes. x and y are ground-based position coordinates, and θ is the target orientation angle. The time index τ refers to a unit of time within the diffusion interval. Once (1) is discretized, τ can simply be thought of as a discrete time index such that a finite number of diffusions will occur between jumps; that number is an exponential random variable.

The derivative needed in solving the Langevin SDE (1) may be computed with a finite difference approximation:

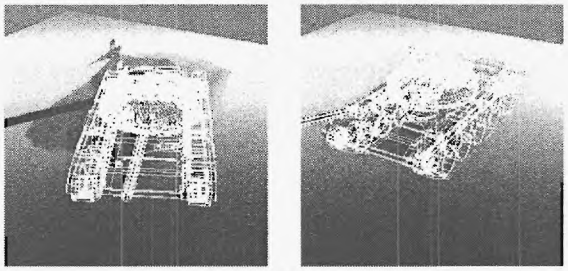
$$\frac{\partial H(c|\mathbf{d})}{\partial c_p} \approx \frac{H(\ldots, c_p + \delta, \ldots|\mathbf{d}) - H(\ldots, c_p - \delta, \ldots|\mathbf{d})}{2\delta}, \tag{2}$$

where c_p is a single parameter of the configuration c, δ is some small deviation of the parameter c_p, and the ellipses indicate the remaining parameters are held fixed.

To summarize this process, the jump-diffusion algorithm starts by estimating an approximate location for a target. If we create a scene by rendering a target at this estimated location, which we will call the hypothesis, we will see that the hypothesized target and corresponding target in the data partially overlap. To refine the initial guess, the diffusion process incrementally adjusts the pose parameters, using the information in the image data. When viewing the hypothesis rendered on top of the data, the overlap between the two improves as the estimated pose parameters converge to their true values. See Figures 5 and 6 for examples of the diffusion process.

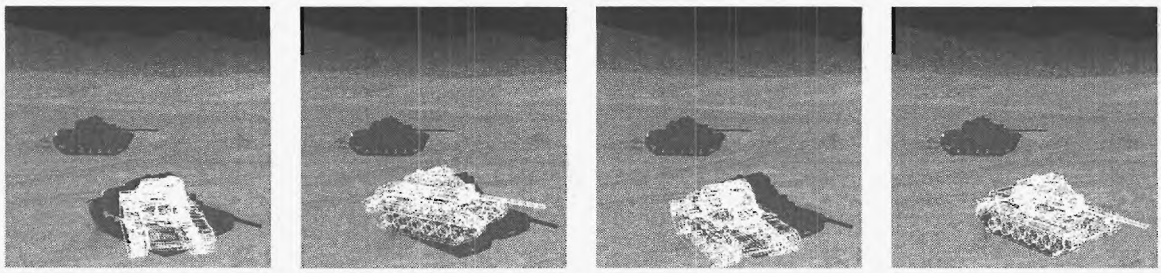
Problems with previous diffusion algorithms: When simulating the Langevin SDE, two issues arise: the choice of stepsizes for the derivative computation and the

[1] U. Grenander and M.I. Miller, "Representations of Knowledge in Complex Systems," *Journal of the Royal Statistical Society, Series B*, Vol. 56, No. 4, pp. 549–603, 1994. Anyone first approaching Grenander's theory should begin with this paper.



(a) (b)

Figure 5: Images (a) and (b) contain a sample, synthetic, noisy, LADAR image. A T62 tank sits at the origin of a ground plane. If we assume that our ATR algorithm initially detects a target in the general vicinity of the T62, we may find that our hypothesized T62, denoted by the wire frame outline, does not overlap perfectly with the T62 in the data. This is shown in (a). After a few iterations of the diffusion process, the pose parameters of the hypothesized T62 should match those of the data T62, and the hypothesized T62 should perfectly overlap with the data image, as shown in (b).



(a) (b) (c) (d)

Figure 6: Images (a), (b), and (c) show a hypothesized target rendered over sample FLIR image data. In each of these images, the estimated pose parameters do not match the true values for the corresponding target within the data image, but there is some overlap. After a sufficient number of iterations of the diffusion process, the estimated pose parameters will adjust until there is a closer match with the data, as shown in image (d).

choice of stepsizes for the discretized diffusions themselves. In the jump-diffusion experiments, both of these were determined empirically through a trial-and-error approach that yielded the best adjustments to the configuration parameters. Ideally, these should be automatically determined, but this is problematic because they depend on the types of targets in the scene, the scene's viewing parameters, and the data likelihood function.

Also, Langevin-style diffusions are more natural when there is an analytic solution to the likelihood derivative. As defined in the ATR problem, this derivative must be approximated with the finite difference equation shown in (2). The logposterior H is effectively a function of an image, and we are taking the derivative of this function with respect parameters used to generate the image. The nature of this approximation demonstrates another reason why determining ideal stepsizes for $d\tau$ and dW_N from (1) is not intuitive.

Lastly, diffusions of this form may lead to crude approximations to the pose of detected targets. In some cases, the Langevin diffusion may not converge, but instead oscillate among values close to the true target pose. These characteristics are common to Langevin-style diffusions.[2]

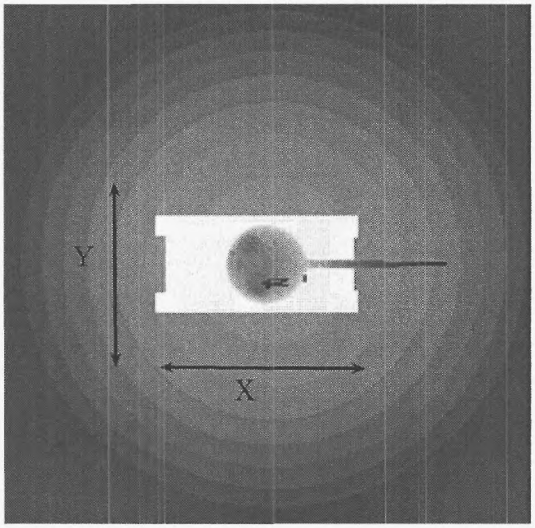
The following discussion will examine our new diffusion algorithm, which redefines the pose parameter refinement problem in way that is more natural to implement and less reliant on arbitrary stepsizes.

New pixel-based diffusion algorithm: In our new diffusion algorithm, we note that the Langevin SDE is not necessary to determine appropriate changes in the coordinate pose parameters (x and y ground plane positions) because those parameters are naturally discretized by the inter-pixel spacing within the image of interest, represented by some real-world unit of measure. For example, if the spacing between neighboring pixels happens to be on the order of centimeters, adjustments to the pose parameters on the order of millimeters will usually not result in a visible change to the hypothesized image. The possibility of subpixel refinement exists, but this requires further study.

The process begins by choosing a *pixel radius*, the desired number of adjacent pixels to consider when determining an updated set of pose parameters for the hypothesized target. A larger pixel radius implies that a larger search space will be considered, increasing the computation time per iteration but decreasing the number of iterations necessary to reach the optimal pose values. Once a pixel radius is chosen, the algorithm proceeds by adjusting the pose parameters for the hypothesized target, selecting values that would result in the image of the hypothesized target moving by a single pixel, or multiple pixels, across the projected surface of the image (see Figure 7). The mapping from pixel coordinates to position coordinates is a function that is built into the OpenGL rendering system we use pattern-theoretic ATR.

In addition to varying the coordinate parameters, we also must consider the orientation parameter θ. This parameter represents the ground-based rotation angle of the target of interest. Rotations do not fit our pixel-based adjustment model as well as changes to the position coordinates, so we are left with choosing a method that will appropriately sample small deviations of θ. This effectively rotates the hypothesized

[2]P.J. Green, "A Primer on Markov chain Monte Carlo," *Complex Stochastic Systems*, Eindhoven, pp. 1–62, 2001.



(a)

(-13.67, 13.67)	(-6.83, 13.67)	(0.00, 13.67)	(6.83, 13.67)	(13.67, 13.67)
(-13.67, 6.83)	(-6.83, 6.83)	(0.00, 6.83)	(6.83, 6.83)	(13.67, 6.83)
(-13.67, 0.00)	(-6.83, 0.00)	(0.00, 0.00)	(6.83, 0.00)	(13.67, 0.00)
(-13.67, -6.83)	(-6.83, -6.83)	(0.00, -6.83)	(6.83, -6.83)	(13.67, -6.83)
(-13.67, -13.67)	(-6.83, -13.67)	(0.00, -13.67)	(6.83, -13.67)	(13.67, -13.67)

(b)

Figure 7: In image (a), a tank is located at some ground-based position that we denote (0.00, 0.00). The tank is imaged by a LADAR sensor positioned 100 m away, with a 30° field-of-view angle, pointing directly toward the tank. If a pixel radius of two were chosen for the diffusion, the 24 pixels surrounding the origin pixel would be selected as candidates for the first iteration. For the image as specified in (a), the 24 test pixels and origin pixel will correspond to the grid of (x, y) pose parameters, in centimeters, found in (b). In this case, the space between each pixel is approximately 6.83 cm.

targets by small amounts in both directions. We have found that specifying a *sweep angle* ϕ_s and *sweep angle stepsize* $d\phi_s$ is flexible enough to allow the algorithm to converge to pose values matching those of the corresponding target in the data, under many varying conditions.

Once the candidate coordinate positions have been chosen, the posterior probabilities are computed for each candidate. The best candidate is considered to be the choice that maximizes the logposterior around the chosen samples. The best candidate is accepted with probability $\beta(c_{curr}, c_{next})$, calculated using the Metropolis-Hastings approach:

$$\beta(c_{curr}, c_{next}) = \min\left(\frac{\pi(c_{next})r(c_{next}, c_{curr})}{\pi(c_{curr})r(c_{curr}, c_{next})}, 1\right). \quad (3)$$

The term c_{next} is the proposed state of the configuration, while c_{orig} is the current state. The functions $r(c_{next}, c_{curr})$ and $r(c_{curr}, c_{next})$ are the transition probabilities.[3]. The function $\pi(c)$ is the probability of being in state c, which in this case is derived from the logposterior H. When selecting candidates during an iteration, candidate position and orientation values can be adjusted by adding a noise term dW_N, just as was done in the Langevin SDE approach, to reduce the possibility of becoming trapped in one of the posterior distribution's local maximums.

A summary of the diffusion algorithm is shown in Table 1.

Table 1: Pixel-based diffusion algorithm.

Initialization

Determine initial pose parameters $c_{next} \leftarrow [x_0, y_0, \theta_0]$

$\leftarrow [x_0, y_0, \theta_0]$

Set pixel radius r_p

Set sweep angle ϕ_s and stepsize $d\phi_s$

Iteration

Repeat

(1) Assign $c_{curr} \leftarrow c_{next}$

(2) Find the set of pixels $\mathcal{R}_p = p_{xy} : \|p_{xy} - p_{x_0y_0}\| < r_p$

(3) Find the $N_\phi = \frac{\phi_s}{d\phi_s}$ angular sweep steps in the set $\mathcal{N}_\phi$

(4) For all pose parameter coordinates in the space $\mathcal{R}_p \times \mathcal{N}_\phi$

(a) Compute the logposterior probability H

(5) Assign c_{next} to the $[x, y, \theta]$ that maximizes H over the space $\mathcal{R}_p \times \mathcal{N}_\phi$

(6) Accept c_{next} with probability $\beta(c_{curr}, c_{next})$

Until $c_{curr} = c_{next}$

---

[3]See A.D. Lanterman, "Jump-diffusion algorithm for multiple target recognition using laser radar range data," *Optical Engineering*, Vol. 40, No. 8, pp. 1724–1728, 2001, for details.

2.1.2 Calibration

The original algorithms and techniques for pattern-theoretic ATR were evaluated on synthetic image sets. These image sets were created either directly from known thermal intensity values or indirectly by generating thermal intensity values probabilistically from a set of known thermal intensity values. This was appropriate for testing the effectiveness of the algorithm, but it does not reflect how the algorithm will work on real thermal infrared imagery. Therefore, the calibration problem needs to be addressed. Suppose that the relationship between the model and the image is affine and can be summarized as

$$\Lambda_{i,t} = a\lambda_{i,t} + b, \tag{4}$$

where $\Lambda_{i,t}$ is the intensity for target t at intensity region i in the image acquired by the FLIR sensor, a and b are the calibration coefficients, and $\lambda_{i,t} = \sum_j \alpha_{j,t}\Phi_{ij,t} + m_{i,t}$. The technique for estimating expansion coefficients presented in Appendix B can be expanded to include the additional calibration terms. This now creates a nonlinear relationship among the parameters to be investigated, so a new solution must be derived. The analysis follows the previous derivation for the logposterior for pixels on target in terms of the expansion coefficients given in Appendix B, except here we include the calibration coefficients a and b as well. This new logposterior may be written as

$$H(\boldsymbol{\alpha}, a, b|D) = -\sum_t \sum_i \sum_{k\in R_{i,t}} \frac{(\Lambda_{i,t} - d(k))^2}{2\sigma^2} - \sum_t \sum_j \frac{\alpha_{j,t}^2}{2\gamma_{j,t}} \tag{5}$$

$$= -\sum_t \sum_i \sum_{k\in R_{i,t}} \frac{\Lambda_{i,t}^2 - 2\Lambda_{i,t}d(k) + d^2(k)}{2\sigma^2} - \sum_t \sum_j \frac{\alpha_{j,t}^2}{2\gamma_{j,t}} \tag{6}$$

$$= -\sum_t \sum_i \frac{N_{i,t}\Lambda_{i,t}^2 - 2\Lambda_{i,t}\sum_{k\in R_{i,t}} d(k) + \sum_{k\in R_{i,t}} d^2(k)}{2\sigma^2} - \sum_t \sum_j \frac{\alpha_{j,t}^2}{2\gamma_{j,t}} \tag{7}$$

$$= -\sum_t \sum_i \frac{N_{i,t}\Lambda_{i,t}^2 - 2\Lambda_{i,t}D_{i,t} + D_{i,t}}{2\sigma^2} - \sum_t \sum_j \frac{\alpha_{j,t}^2}{2\gamma_{j,t}}. \tag{8}$$

Incorporating $\Lambda_{i,t} = a\lambda_{i,t} + b$ where $\lambda_{i,t} = \sum_j \alpha_{j,t}\Phi_{ij,t} + m_{i,t}$, we must take the derivatives of $H(\boldsymbol{\alpha}, a, b|D)$ with respect to each $\alpha_{j,t}$, a, and b. To make the following derivations cleaner, we begin by mentioning following derivatives:

$$\frac{\partial}{\partial\alpha_{j,t}}\Lambda_{i,t'} = a\frac{\partial}{\partial\alpha_{j,t}}\lambda_{i,t'} = \begin{cases} a\Phi_{ij,t} & \text{if } t = t' \\ 0 & \text{if } t \neq t' \end{cases} \tag{9}$$

$$\frac{\partial}{\partial a}\Lambda_{i,t} = \lambda_{i,t} \tag{10}$$

$$\frac{\partial}{\partial b}\Lambda_{i,t} = 1. \tag{11}$$

We continue with the derivation by computing the derivatives of $H(\boldsymbol{\alpha}, a, b|D)$ with respect to each $\alpha_{j,t}$:

$$\frac{\partial H}{\partial \alpha_{j,t}} = -\sum_{t'}\sum_{i} \frac{2N_{i,t'}\Lambda_{i,t'}\frac{\partial}{\partial \alpha_{j,t}}\Lambda_{i,t'} - 2\frac{\partial}{\partial \alpha_{j,t'}}\Lambda_{i,t'}D_{i,t'}}{2\sigma^2} - \frac{\alpha_{j,t}}{\gamma_{j,t}} \tag{12}$$

$$= -\sum_{i} \frac{N_{i,t}\left[a\left(\sum_k \alpha_{k,t}\Phi_{ik,t} + m_{i,t}\right) + b\right] a\Phi_{ij,t} - a\Phi_{ij,t}D_{i,t}}{\sigma^2} - \frac{\alpha_{j,t}}{\gamma_{j,t}} \tag{13}$$

$$= -\sum_{i} \frac{\left(aN_{i,t}\sum_k \alpha_{k,t}\Phi_{ik,t} + aN_{i,t}m_{i,t} + N_{i,t}b\right) a\Phi_{ij,t} - a\Phi_{ij,t}D_{i,t}}{\sigma^2} - \frac{\alpha_{j,t}}{\gamma_{j,t}} \tag{14}$$

$$= -\frac{\sum_i a^2\Phi_{ij,t}N_{i,t}\sum_k \alpha_{k,t}\Phi_{ik,t} + \sum_i \Phi_{ij,t}\left(a^2N_{i,t}m_{i,t} + abN_{i,t} - aD_{i,t}\right)}{\sigma^2} - \frac{\alpha_{j,t}}{\gamma_{j,t}} \tag{15}$$

$$= -\frac{a^2}{\sigma^2}\sum_k \alpha_{k,t}\sum_i N_{i,t}\Phi_{ik,t}\Phi_{ij,t} + \frac{a^2}{\sigma^2}\sum_i \Phi_{ij,t}\left(N_{i,t}m_{i,t} + \frac{bN_{i,t}}{a} - \frac{D_{i,t}}{a}\right) - \frac{\alpha_{j,t}}{\gamma_{j,t}} \tag{16}$$

To maximize the logposterior, we must satisfy these $\sum_t J_t$ necessary conditions:

$$-\frac{a^2}{\sigma^2}\sum_k \alpha_{k,t}\sum_i N_{i,t}\Phi_{ik,t}\Phi_{ij,t} + \frac{a^2}{\sigma^2}\sum_i \Phi_{ij,t}\left(N_{i,t}m_{i,t} + \frac{bN_{i,t}}{a} - \frac{D_{i,t}}{a}\right) - \frac{\alpha_{j,t}}{\gamma_{j,t}} = 0,\ \forall j,t \tag{17}$$

$$\sum_k \alpha_{k,t}\sum_i N_{i,t}\Phi_{ik,t}\Phi_{ij,t} - \sum_i \Phi_{ij,t}\left(N_{i,t}m_{i,t} + \frac{bN_{i,t}}{a} - \frac{D_{i,t}}{a}\right) + \frac{\sigma^2\alpha_{j,t}}{a^2\gamma_{j,t}} = 0,\ \forall j,t \tag{18}$$

Fortunately these are T sets of linear equations, one set for each target, conveniently expressed in matrix form:

$$\left(\begin{bmatrix} \sum_i N_{i,t}\Phi_{i1,t}^2 & \cdots & \sum_i N_{i,t}\Phi_{iJ,t}\Phi_{i1,t} \\ \vdots & & \vdots \\ \sum_i N_{i,t}\Phi_{i1,t}\Phi_{iJ,t} & \cdots & \sum_i N_{i,t}\Phi_{JJ,t}^2 \end{bmatrix} + diag\left(\begin{bmatrix} \frac{\sigma^2}{a^2\gamma_{1,t}} \\ \vdots \\ \frac{\sigma^2}{a^2\gamma_{J,t}} \end{bmatrix}\right)\right)\begin{bmatrix} \alpha_{1,t} \\ \vdots \\ \alpha_{J,t} \end{bmatrix}$$
$$= \begin{bmatrix} \sum_i \Phi_{i1,t}\left(\frac{D_{i,t}-bN_{i,t}}{a} - N_{i,t}m_{i,t}\right) \\ \vdots \\ \sum_i \Phi_{iJ,t}\left(\frac{D_{i,t}-bN_{i,t}}{a} - N_{i,t}m_{i,t}\right) \end{bmatrix}. \tag{19}$$

If we keep the $\alpha_{j,t}$ terms constant, we can find the derivatives of $H(\boldsymbol{\alpha}, a, b|D)$ with respect to a and b and maximize the logposterior with respect to these terms. The

derivative of $H(\boldsymbol{\alpha}, a, b|D)$ with respect to a is

$$\frac{\partial H}{\partial a} = -\sum_t \sum_i \frac{2N_{i,t}\Lambda_{i,t}\frac{\partial}{\partial a}\Lambda_{i,t} - 2\frac{\partial}{\partial a}\Lambda_{i,t}D_{i,t}}{2\sigma^2} \tag{20}$$

$$= -\sum_t \sum_i \frac{N_{i,t}\Lambda_{i,t}\lambda_{i,t} - \lambda_{i,t}D_{i,t}}{\sigma^2} \tag{21}$$

$$= -\sum_t \sum_i \frac{N_{i,t}\left(a\lambda_{i,t} + b\right)\lambda_{i,t} - \lambda_{i,t}D_{i,t}}{\sigma^2} \tag{22}$$

$$= -\sum_t \sum_i \frac{aN_{i,t}\lambda_{i,t}^2 + bN_{i,t}\lambda_{i,t} - \lambda_{i,t}D_{i,t}}{\sigma^2} \tag{23}$$

$$= -\frac{a}{\sigma^2}\sum_t \sum_i N_{i,t}\lambda_{i,t}^2 - \frac{b}{\sigma^2}\sum_t \sum_i N_{i,t}\lambda_{i,t} + \frac{1}{\sigma^2}\sum_t \sum_i \lambda_{i,t}D_{i,t} \tag{24}$$

and the derivative of $H(\boldsymbol{\alpha}, a, b|D)$ with respect to b is

$$\frac{\partial H}{\partial b} = -\sum_t \sum_i \frac{2N_{i,t}\Lambda_{i,t}\frac{\partial}{\partial b}\Lambda_{i,t} - 2\frac{\partial}{\partial b}\Lambda_{i,t}D_{i,t}}{2\sigma^2} \tag{25}$$

$$= -\sum_t \sum_i \frac{N_{i,t}\Lambda_{i,t} - D_{i,t}}{\sigma^2} \tag{26}$$

$$= -\sum_t \sum_i \frac{N_{i,t}\left(a\lambda_{i,t} + b\right) - D_{i,t}}{\sigma^2} \tag{27}$$

$$= -\sum_t \sum_i \frac{aN_{i,t}\lambda_{i,t} + bN_{i,t} - D_{i,t}}{\sigma^2} \tag{28}$$

$$= -\frac{a}{\sigma^2}\sum_t \sum_i N_{i,t}\lambda_{i,t} + \frac{b}{\sigma^2}\sum_t \sum_i N_{i,t} - \frac{1}{\sigma^2}\sum_t \sum_i D_{i,t}. \tag{29}$$

Two maximize this part of the posterior, the derivatives must satisfy these respective conditions:

$$a\sum_t \sum_i N_{i,t}\lambda_{i,t}^2 + b\sum_t \sum_i N_{i,t}\lambda_{i,t} = \sum_t \sum_i \lambda_{i,t}D_{i,t} \tag{30}$$

$$a\sum_t \sum_i N_{i,t}\lambda_{i,t} + b\sum_t \sum_i N_{i,t} = \sum_t \sum_i D_{i,t}. \tag{31}$$

It is immediately apparent that these conditions can be represented in a form of a matrix to solve for a and b. Also note that these set of equations represent the least squares solution to the problem of determining the affine parameters a and b that best fit the derived thermal intensities $\lambda_{i,t}$ from the data intensities $d(k)$.

We now have two sets of equations: one set that maximizes the logposterior H with respect to the α terms when the a and b terms are held constant and another set that maximizes the same logposterior with respect to the a and b terms when the α terms are held constant. Written together, these sets of equations represent a system of nonlinear equations. Many techniques exist to find solutions to such a system, but

these techniques can be difficult to implement for various reasons. Since this system separates so nicely into two systems of linear equations, it is reasonable to suppose that there may be a way to view the problem in terms of iteratively solving these linear equations. If we can determine a "good" initial value, we may consider the algorithm shown in Table 2 as a guide to iteratively estimate the expansion terms and calibration coefficients.

Table 2: Algorithm to compute the expansion coefficients α_j and thermal calibration coefficients a and b.

Initialization

Compute $N_{i,t}$ and $D_{i,t}$ $\forall$ i, t
Assign $\alpha_{j,t}^{(n)} \leftarrow 0$ $\forall$ j, t
Assign $a^{(n)} \leftarrow 1$
Assign $b^{(n)} \leftarrow 0$

Iteration

Repeat
(1) Assign $a^{(n-1)} \leftarrow a^{(n)}$
(2) Assign $b^{(n-1)} \leftarrow b^{(n)}$
(3) Assign $\alpha_{j,t}^{(n-1)} \leftarrow \alpha_{j,t}^{(n)}$ $\forall$ j, t
(4) Solve for the $\alpha_{j,t}^{(n)}$ coefficients using $a^{(n-1)}$ and $b^{(n-1)}$ $\forall$ j, t
(5) Compute $\lambda_{i,t} = \sum_j \alpha_{j,t}^{(n)} \Phi_{ij,t} + m_{i,t}$ $\forall$ i, t
(6) Solve for $a^{(n)}$ and $b^{(n)}$ using $\lambda_{i,t}$, $N_{i,t}$, and $D_{i,t}$ $\forall$ i, t
Until $\left\| [\boldsymbol{\alpha}, a, b]^{(n)} - [\boldsymbol{\alpha}, a, b]^{(n-1)} \right\| < \epsilon$

2.2 ATR with 3-D Data

The exploitation of 3-D has garnered tremendous attention by DARPA and the Air Force. Different algorithms have been proposed by some of the usual suspects, such as MITRE, Alphatech, and Carnegie Mellon. These algorithms have generally involved extraction of features to achieve real-time performance constraints demanded by DARPA's short-term goals. Our interest in the problem runs deeper. Instead of developing yet another real-time algorithm to compete in a shootout with the myriad real-time algorithms that have already been proposed, we are addressing the more fundamental question: what is the best that we could do with data of a particular quality, independent of whatever particular algorithm is used? As in our infrared work, we bypass the feature extraction stage and conduct inference directly from the full available data, since information may be lost in the feature extraction stage. If a feature-based algorithm manages to achieve our theoretical lower bounds, that implies the features chosen constitute sufficient statistics for the inference problem.

We have developed statistical likelihood models for different kinds of 3-D data. For raw laser radar imagery, we employ models based on the underlying physics of the detector. For assembled "point clouds," whose statistics are a complex interaction of the underlying sensor characteristics and the algorithms used to assemble multiple views, we consider a Poisson point process model chosen for its analytical tractability. Our prime emphasis in this work has been on the creation of performance bounds, as described in Sections 2.3.2 and 2.3.3.

The improved "pixel-based" diffusions we developed for infrared data, described Section 2.1.1, are also applicable to jump-diffusion algorithms for target recognition with laser radar data.[4]

2.3 Performance Bounds via Kullback-Leibler Distances, Chernoff Distances, and Fisher Information

We have developed generic performance bounds, based on Stein's lemma, Chernoff bounds, and Fisher information, which aim to be independent of any particular ATR algorithm. Much of this work follows the lines of theory developed by Anuj Srivastava, Ulf Grenander, and Michael Miller.

2.3.1 Performance Bounds with Low Frequency Radar Data

Inspired by the AFOSR DURIP project titled "Integrated Sensing and Computation for Passive Covert Radar, Signals Intelligence, and Other Applications Driven by Moore's Law," we have demonstrated the power of our information-theoretic performance bounds on the specific application of ATR with passive radar data by comparing our predicted performance measures with empirical performance metrics derived from Monte Carlo runs. Appendix D discusses computing such Chernoff bounds based on a Rician model appropriate for low-frequency radar. Such bounds allow us to answer questions such as "how long must we collect data on an aircraft before we can make an recognition decision with a certain probability of correct decision?"

As described in detail in the next section, the Kullback-Leibler distance provides another metric for detection problems. Appendix E presents an approximation of the Kullback-Leibler distance for Rician models, and Appendix I applies this line of reasoning to low-frequency radar.

2.3.2 Performance Bounds with 3-D Point Cloud Data

At the 2005 AFOSR program review in Raleigh, NC, we presented results on performance analysis using Fisher Information and Kullback-Leibler distances for 3-D data using our Poisson "point cloud" model. Our goal here is to formulate likelihood models on point-cloud data, and not features extracted from those point clouds.

This work was inspired by reading the description of DARPA's E3D program, which sought "efficient techniques for rapidly exploiting 3-D sensor data to precisely locate

[4] A. D. Lanterman, "Jump-diffusion Algorithm for Multiple Target Recognition using Laser Radar Range Data," *Optical Engineering*, Vol. 40, No. 8, pp. 1724–1728, 2001.

and recognize targets." The BAA for it contained various demands for different stages of the program, such as "The Target Acquisition and Recognition technology areas will develop techniques to locate and recognize articulating, reconfigurable targets under partial obscuration conditions, with an identification probability of 0.85, a target rejection rate less than 5%, and a processing time of 3 minutes per target or less." This naturally leads to some questions: 1) If such a milestone is not reached, *is that the fault of the algorithm or the sensor?* 2) What performance from a particular sensor is necessary to achieve a certain level of ATR performance, *independent of the question of what algorithm is used?*

Theoretical lower bounds on algorithm performance give algorithm designers a goal to shoot for, and provider criteria for various sensor hardware tradeoffs. In a similar vein, ATR algorithms are typically designed under current computational hardware constraints; however, computers keep getting faster, so it makes sense to ask what ultimate underlying limits are placed by the sensor hardware. The goal is to understand the information content available in the data itself, instead of skipping straight to algorithm development.

Grenander's Pattern Theory provides our philosophical framework. Most 3-D data ATR algorithms (and ATR algorithms with other kinds of data) seek features that are invariant to pose (position and orientation). In contrast, the Grenander approach does not hide from the pose parameter, and explicitly co-estimates it or integrates it out. At a given viewing angle, Target A at one orientation may look much like Target B at a different orientation. As Grenander, Miller, and Srivastava note,[5] "the nuisance parameter of orientation estimation plays a fundamental role in determining the bound on recognition."

A "true" statistical model, which exactly matches the complex interactions of the sensors and Jigsaw-like software, is probably analytically intractable (if it could be developed at all). We base our models on inhomogeneous Poisson processes, since they possess many convenient properties. Although the real data will be distributed more uniformly than a Poisson distribution will predict, the Poisson-based likelihoods should provide useful results. Since modern 3-D point clouds will result from the assembly of multiple views, solely modeling range measurement error across a single line of sight is be insufficient. We assume that the points are seen with an additive Gaussian error of appropriate covariance. This corresponds to a "translated Poisson process," where we essentially assume that the intensity of the observed Poisson process is given by visible portion of the shell of the model convolved with a "point spread function" defined by the measurement covariance.

The experiments described in this section employed four target models, shown in Figure 8, taken from the AFRL 3D Challenge Problem, which was distributed on DVDs at the 2003 SPIE Defense & Security Symposium. The performance graphs in the remaining subsections plot various metrics vs. the standard deviation of the measurement errors. A circularly symmetric error density is assumed.

The DVDs for the 2003 Challenge Problem only have five different look angles per

[5]U. Grenander, M.I. Miller, and A. Srivastava, "Hilbert-Schmidt Lower Bounds for Estimators on Matrix Lie Groups for ATR," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 20, No. 2, Aug. 1998, pp. 790–802.

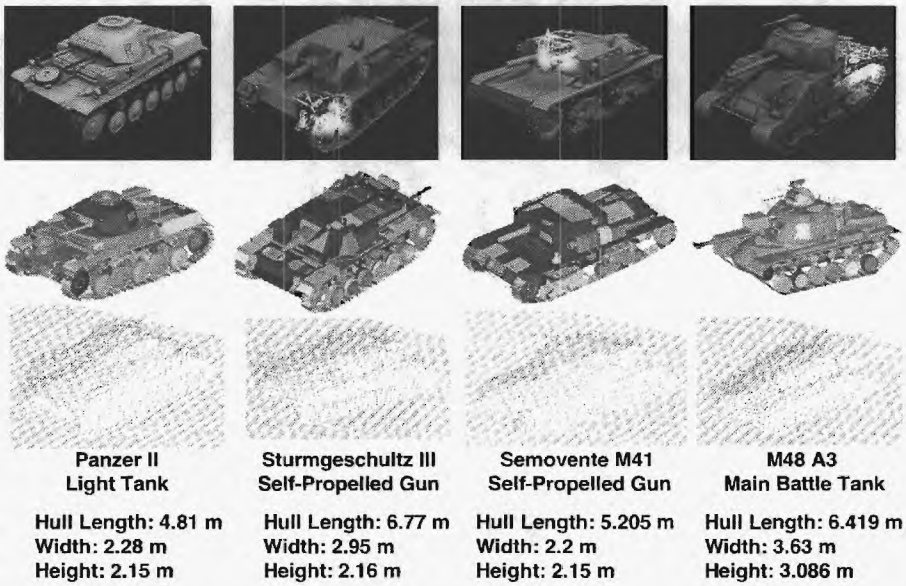


Figure 8: Four targets from the AFRL 2003 3D Challenge Problem, used in our 3-D point-cloud performance model experiments.

target. Hence, we computed bounds for one particular look angle, using adjacent look angles to compute derivatives with respect to orientation. The computed information metrics will, in general, be a function of the orientation, since targets will look different to the sensor at different orientations (in particular, some parts of the target may be obscured if no views are available from a certain range of angles.) The limitations of this Challenge Problem data set was one of the motivating factors of the development of our new simulator described in Appendix A.

Cramér-Rao bounds for point-cloud data: Cramér-Rao bounds for continuous pose parameters are given by the diagonal elements of the inverse of the Fisher information matrix. Cross-terms show how estimate errors are correlated. Figures 9, 10, and 11 illustrate Cramér-Rao bounds on the pose parameters of x-position, y-position, and orientation angle, respectively, for the Sturmgeschultz and Semovente targets. We assume that the target is sitting on a flat surface, and hence the z-position is known. For each target type, two lines are given. One line corresponds to performance in an artificial case where the all of the parameters are known except the target of interest. The other shows performance in the case where all three parameters need to be co-estimated. The distance between the lines shows the "performance hit" that results from the coupling of the parameters.

We emphasize that even if a rather accurate model is available, we would not expect a real system to achieve the bound in practice, since there will always be effects in the real data not present in the model. In addition, in this particular case of 3-D point cloud data, our inhomogenous Poisson model was formulated for computational convenience, not fidelity to reality. We hope, however, that the bounds can provide insight into

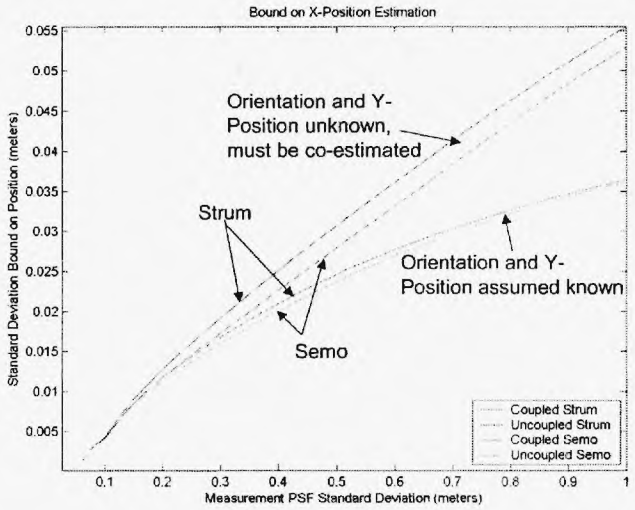


Figure 9: Cramér-Rao bounds on the x-position pose parameter of a Sturmgeschulz and Semovente as a function of measurement error.

overall performance trends. In practice, it may be feasible to vary the overall intensity of the Poisson process to bring the bounds into accordance with results from Monte Carlo runs.

I-divergence metrics for point-cloud data: Likelihood ratios are the key statistics in hypothesis testing problems such as automatic target recognition. The Kullback-Leibler distance, also known as the relative entropy, is the expected value of the likelihood ratio, assuming the data is generated according to the "alternative hypothesis" (in the "alternative" vs. "null" nomenclature). According to Stein's lemma, the Kullback-Leibler distance drives the asymptotic performance of the detection problem. For Gaussian data, the Kullback-Leibler distance reduces to a squared-error metric. For Poisson data, as assumed in our point-cloud data model, the Kullback-Leibler distance reduces to Csiszár's I-divergence.

When computing the I-divergence to compare targets, we adjust the pose of the "alternate" target to get closest match to the "true" target as seen by the sensor system. The work of Grenander, Srivastava, and Miller[6] shows that although hypothesis testing with nuisance parameters (in this case) will be dominated by this I-divergence term, there will also be a second term involving the Cramér-Rao bound on the nuisance parameters. This notion links estimation and recognition performance, and connects the experiments described this subsection with the those described in the previous subsection. Here, we only consider the effect of the "first term," i.e. the I-divergence. We hope to study the inclusion of the second term in future work.

Figures 12, 13, 14, and 15 plot the I-divergence metrics (lower I-divergence indicates greater difficulty of discrimination) between the targets. Each of the four plots takes a

[6]U. Grenander, A. Srivastava, and M.I. Miller, "Asymptotic Performance Analysis of Bayesian Target Recognition," *IEEE Trans. Information Theory*, Vol. 46, No. 4, July 2000, pp. 1658–1665.

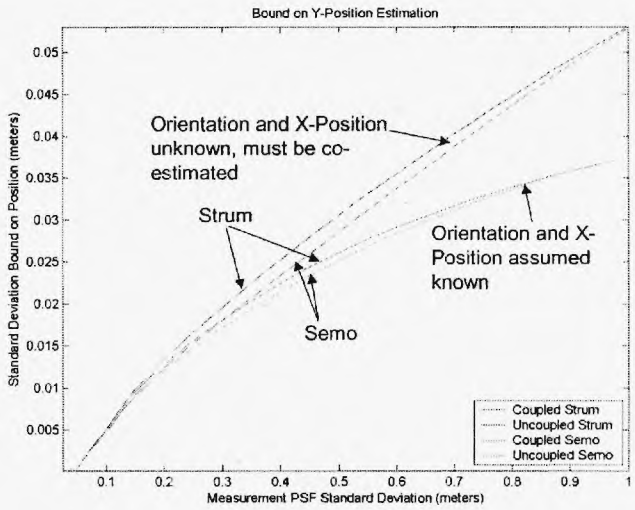


Figure 10: Cramér-Rao bounds on the y-position pose parameter of a Sturmgeschulz and Semovente as a function of measurement error.

single target type to be the "alternative" hypothesis; each line within a plot represents a different "null" hypothesis.

Interestingly, the I-divergence is *not symmetric* in general; for instance, the $I(M48\|panzer)$ line in Figure 15 is different than the $I(panzer\|M48)$ line in Figure 12. This is why we were sure to make a distinction between the "alternative" and "null" hypothesis in the preceeding paragraphs. During the question & answer discussion period after Aaron Lanterman's closing talk at the *ARO/ARMDEC Workshop on Information-Theoretic Image Processing*, Prof. Al Hero of the Univ. of Michigan commented that we should not shy away from this asymmetry or find it unusual; instead, we should "embrace the asymmetry."

Such asymmetry manifests itself in the "pop-out" experiments in psychology.[7] Consider setting a probability of a Type-1 error (usually called "false alarm") in a Neyman-Pearson detection framework. Consider two hypothetical problems, illustrated in Figure 16. In the problem on the left, we want to detect a Panzer in a sea of M48 clutter; in the problem on the right, we want to detect an M48 in a sea of Panzer clutter. It may be counterintuitive, but one problem will be more difficult than the other. Both I-divergence curves for these two target types are shown in Figure 17. Note that the problem illustrated on the left of Figure 16 is easier than that on the right for measurement noise with standard deviation greater than 0.35 meters, whereas the problem illustrated on the right of 16 is easier than that on the left for measurement noise with standard deviation less than 0.35 meters.

[7]A.L. Yuille and J.M. Coughlan, "Fundamental Limits of Bayesian Inference: Order Parameters and Phase Transitions for Road Tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 2, No. 2, Feb. 2000, pp. 160–173; Y.N. Wu, S.C. Zhu, and Z. Liu, "Equivalence of Julesz Ensembles and FRAME Models," Intl J. of Computer Vision, Vol. 38, No. 3, 2000, pp. 247–265; see Section 7 in particular.

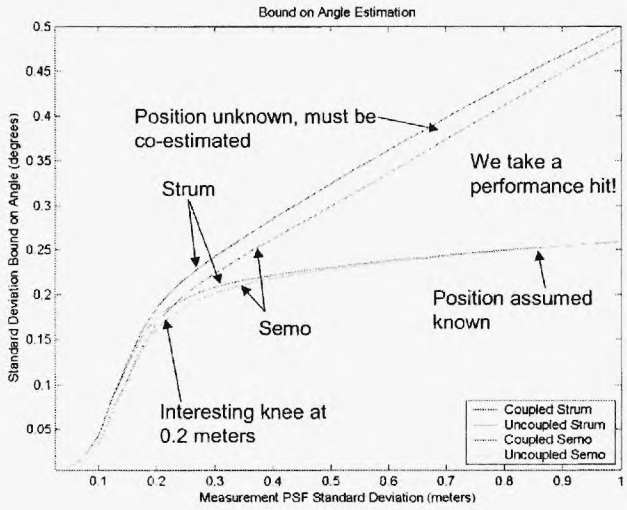


Figure 11: Cramér-Rao bounds on the angle pose parameter of a Sturmgeschulz and Semovente as a function of measurement error.

The cases of Panzer vs. Semovente (Figure 18), and Panzer vs. Sturmgeschultz (Figure 19) have similar crossover points in the 0.3 to 0.4 meter region. The M48 vs. Semovente (Figure 20) and M48 vs. Strumgeschultz cases have lower crossover points, at around 0.2 and 0.1 meters, respectively.

The Semovente vs. Sturmgeschultz case (Figure 22) is particularly interesting, since the two curves lie almost on top of one another; essentially, at any resolution, finding a Semovente in a sea of cluttering Sturmgeschulzen is no more or less difficult than finding a Strumgeschulz in a sea of cluttering Semoventes.

2.3.3 Performance Bounds with Single-Image Laser Radar Data

If we are not trying to use point-cloud data, as in the previous subsection, and are instead single views from a laser radar, more accurate likelihood models may be employed. Appendix C presents results on the computation of Cramér-Rao bounds on pose parameters using such models.

2.4 Finite-Set Statistics for Multitarget Tracking

Inspired by Keith Kastella's work with Joint Multitarget Densities for AFRL, we integrated another research effort (previously supported by startup funds from the School of Electrical Engineering, and primarily supported by funds from the Paris Professorship) on data fusion in multitarget tracking using Ronald Mahler's Finite-Set Statistics, particularly his Probability Hypothesis Densities (PHDs), into this PTATR effort.

Via simulations, we have illustrated the promise of PHD-based multitarget, multisensor tracking using an FM-radio-based passive radar scenario designed to match a system being developed by NATO NC3A. Appendix G addresses some issues that

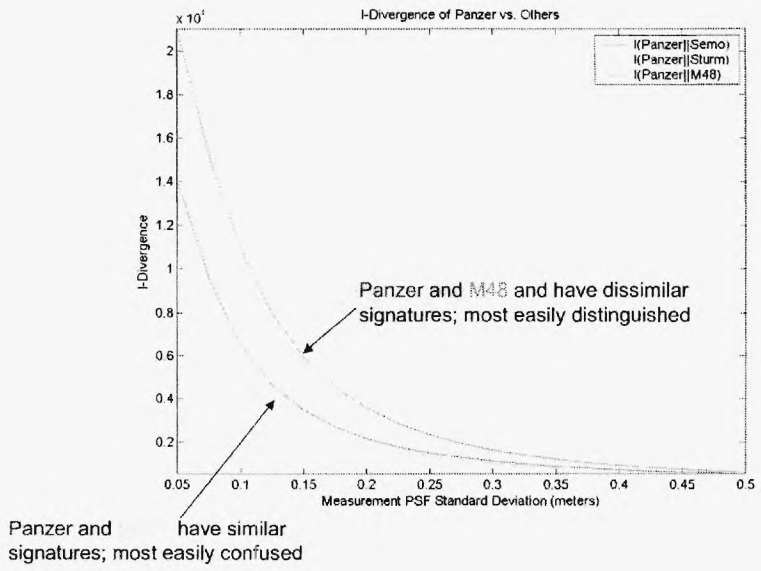


Figure 12: I-divergences between the Panzer model (taken to be the "alternative" hypothesis) and the Semovente, Sturmgeschulz, and Panzer models (taken to be the "null" hypothesis).

have vexed the PHD community, particularly the extraction of peaks (which represent targets) from the PHD, and Appendix H studies the effect of multipath on the algorithm. Our eventual goal is to apply such algorithms to data collected using equipment purchased under the AFOSR DURIP project titled "Integrated Sensing and Computation for Passive Covert Radar, Signals Intelligence, and Other Applications Driven by Moore's Law."

Although our particular example of data fusion and tracking using PHDs happens to be passive radar, the theory is quite general and may be applied to data fusion with other kinds of sensors.

3 Personnel Supported

Jason Dixon, graduate student, fully supported by the AFOSR PTATR grant from August 15, 2004 to August 31, 2006 (41,344.70); support to continue the work past August 2006 provided by the Demetrius T. Paris Professorship. Primary developer of ATR theory and algorithms for target recognition with FLIR and LADAR data, as well as associated applied performance analysis techniques

Lisa Ehrman, graduate student, partially supported by the AFOSR PTATR grant from January 1, 2004 to April 30, 2004 (4,791.08) to focus on performance estimation via Kullback-Leibler distances and Chernoff information measures, with particular application to ATR with low frequency radar. Primarily supported by NATO Consultation, Command and Control Agency (NC3A).

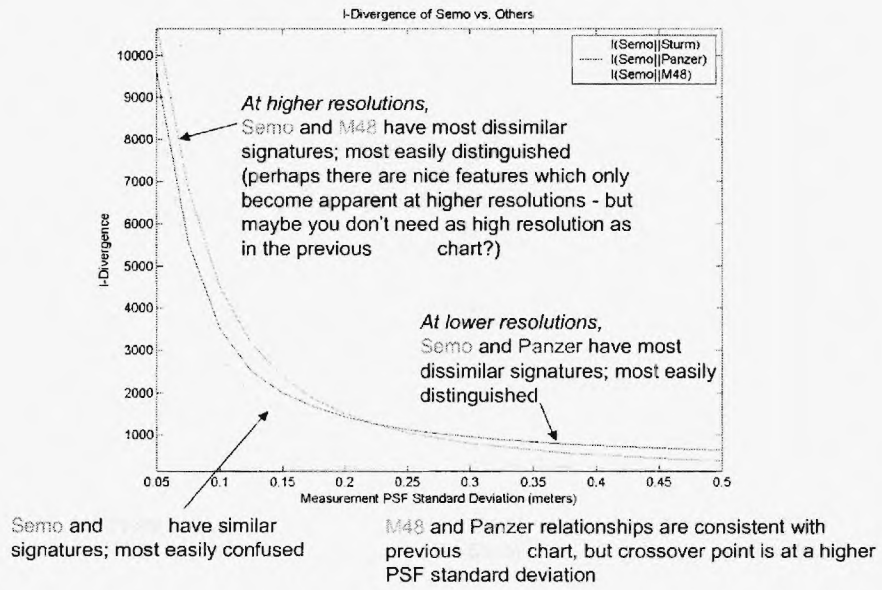


Figure 13: I-divergences between the Semovente model (taken to be the "alternative" hypothesis) and the Sturmgeschulz, Panzer, and M48 models (taken to be the "null" hypothesis).

Linda Fomundam, graduate student, fully supported the AFOSR PTATR grant from August 15, 2004 to December 31, 2004 (6,805.45). Developed code to perform computations of Kullback-Leibler distances and Fisher information from the Poisson "point cloud" model.

Aaron Lanterman, Assistant Professor, some summer salary support from August 1, 2004 to August 15, 2005 (16,026.19). Principal investigator.

Jonathan Morris, graduate student, fully supported by the AFOSR PTATR grant from August 15, 2003 to May 15, 2004 (9,042.16). Began effort to port legacy Silicon Graphics code to the OpenGL platform.

Martin Tobias, graduate student, partially supported by the AFOSR PTATR grant from September 1, 2004 (19,913.23); primarily supported by the startup-funds from the School of Electrical and Computer Engineering and the Demetrius T. Paris Junior Professorship. Also supported by NATO Consultation, Command and Control Agency (NC3A) during a summer internship in 2005. Focused on data fusion via finite-set statistics (similar in spirit to the Keith Kastella's Joint Multitarget Probabilities) for target tracking with passive radar data.

4 Technical Publications

Some of the publications described below will be available from the PTATR (Pattern-Theoretic Automatic Target Recognition) website at `users.ece.gatech.edu/~lanterma/ptatr`.

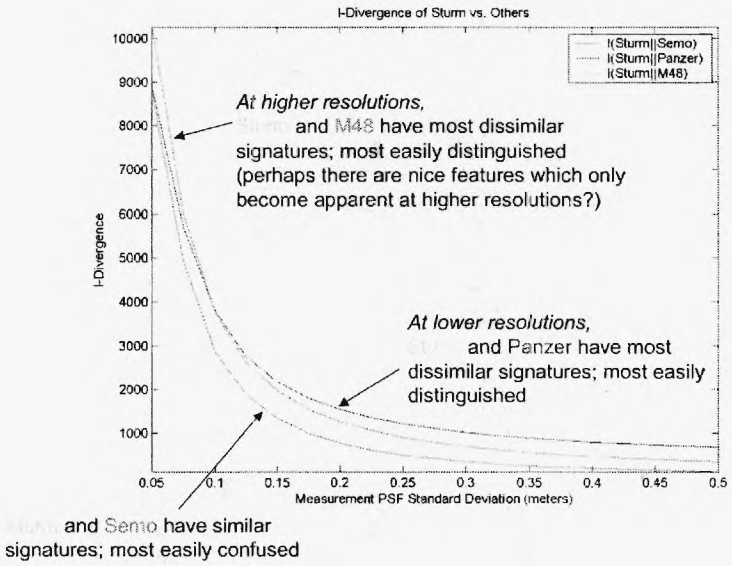


Figure 14: I-divergences between the Sturmgeschulz model (taken to be the "alternative" hypothesis) and the Semovente, Panzer, and M48 models (taken to be the "null" hypothesis).

A select subset, particularly preprints of submitted or to-be-submitted journal papers, which are referred to in other sections of this document, are provided as appendices to this report. Note that later versions of the papers that may appear on the PTATR website or in print may differ than the early drafts here.

4.1 Doctoral Dissertations

Georgia Tech dissertations may be easily obtained from the Electronic Thesis and Dissertation Collection at `etd.gatech.edu`.

J.H. Dixon, *Pattern-Theoretic Automatic Target Recognition for Infrared and Laser Radar Data*, expected to be completed Summer 2007.

L.M. Ehrman, *An Algorithm for Automatic Target Recognition Using Passive Radar and an EKF for Estimating Aircraft Orientation*, Fall 2005.

M. Tobias, *Probability Hypothesis Densities for Multitarget, Multisensor Tracking with Application to Passive Radar*, Spring 2006.

4.2 Journal Publications

L.M. Ehrman and A.D. Lanterman, "Chernoff-Based Prediction of ATR Performance from Rician Radar Data, with Application to Passive Radar," *Optical Engineering*, letter of acceptance subject to minor revision received Feb. 2, 2006; revision submitted Feb. 2007. (Appendix D).

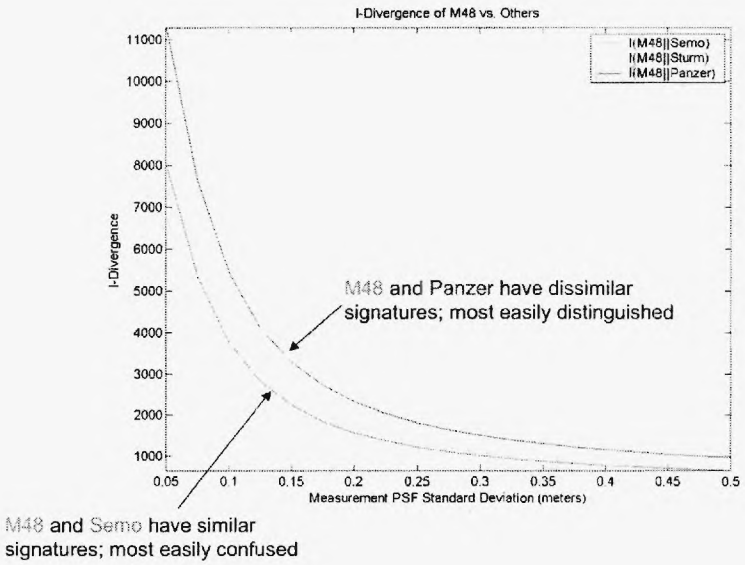


Figure 15: I-divergences between the M48 model (taken to be the "alternative" hypothesis) and the Semovente, Sturmgeschulz, and Panzer models (taken to be the "null" hypothesis).

L.M. Ehrman and A.D. Lanterman, "A Laplace Approximation of the Kullback-Leibler Distance Between Ricean Distributions," *IEEE Trans. on Information Theory*, to be submitted. (Appendix E).

A.D. Lanterman, "Continuous-Time Jump Processes, with Application to Shapes on the Lattice," *Statistics and Computing*, letter requesting major revision received Sept. 2005; manuscript undergoing revision. (Appendix F)

M. Tobias and A.D. Lanterman, "Probability Hypothesis Density-Based Multitarget Tracking with Bistatic Range and Doppler Observations," *IEE Proc. Radar, Sonar, and Navigation*, Vol. 152, No. 3, June 2005, pp. 195–205. (Available from `ieeexplore.ieee.org`, paper number 01459156.)

M. Tobias and A.D. Lanterman, "Techniques for Birth Particle Placement in the PHD Particle Filter, Applied to Passive Radar," *IEE Proc. Radar, Sonar, and Navigation*, submitted April 7, 2007. (Appendix G).

M. Tobias and A.D. Lanterman, "Multipath Effects and the PHD Particle Filter," *IEE Proc. Radar, Sonar, and Navigation*, to be submitted. (Appendix H).

4.3 Conference Publication

L.M. Ehrman and A.D. Lanterman, "Robust Algorithm for Automated Target Recognition using Precomputed Radar Cross Sections, *Automatic Target Recognition XIV*, Proc. SPIE 5426, Ed: F.A. Sadjadi, April 12-16, 2004, pp. 197–208. (Appendix I).

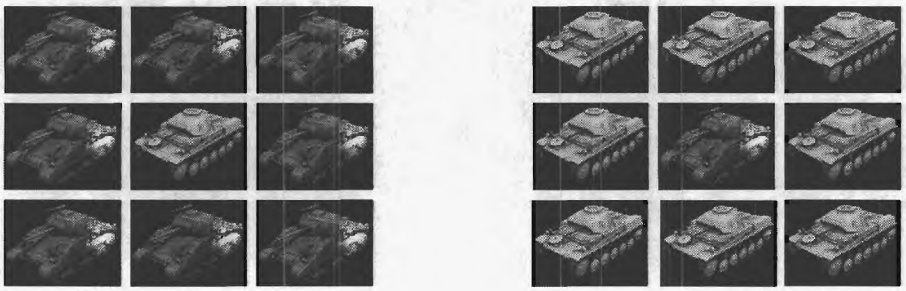


Figure 16: Illustration of two asymmetric "pop out experiments." On the left, we want to detect Panzer among cluttering M48s; on the right, we want to detect an M48 among cluttering Panzers.

A.D. Lanterman, "Passive Radar Imaging and Target Recognition using Illuminators of Opportunity," *NATO Symposium on Target Identification and Recognition Using RF Systems*, Oslo, Norway, Oct. 11-13, 2004.

L.M. Ehrman and A.D. Lanterman, "Assessing the Performance of a Covert Automatic Target Recognition Algorithm," *Automatic Target Recognition XV*, Proc. SPIE 5807, Ed: F.A. Sadjadi, Orlando, FL, March 28-April 1, 2005, pp. 77–87.

J.H. Dixon and A.D. Lanterman, "Toward Practical Pattern-Theoretic ATR Algorithms for Infrared Imagery," *Automatic Target Recognition XVI*, SPIE Vol. 6234, Ed: F.A. Sadjadi, April 2006, pp. 212–220. (Appendix B).

J.H. Dixon and A.D. Lanterman, "Information-Theoretic Bounds on Target Recognition Performance from Laser Radar Data," *Automatic Target Recognition XVI*, SPIE Vol. 6234, Ed: F.A. Sadjadi, April 2006, pp. 394–403. (Appendix C).

5 Interactions/Transitions

5.1 Conference and Workshops

J.H. Dixon, "Toward Practical Pattern-Theoretic ATR Algorithms for Infrared Imagery" and "Information-Theoretic Bounds on Target Recognition Performance from Laser Radar Data" (papers listed above under "Conference Publications"), SPIE Defense and Security Symposium, Orlando, FL, April 17-21, 2006 (1,184.04). A.D. Lanterman also attended (1,003.47).

A.D. Lanterman, "Passive Radar Imaging and Target Recognition using Illuminators of Opportunity," *NATO Symposium on Target Identification and Recognition Using RF Systems*, Oslo, Norway, Oct. 11-13, 2004 (trip paid for by NATO).

A.D. Lanterman, "General Pattern Theory," *AFRL ATR Theory MURI Workshop*, Dayton, OH, Dec. 1, 2004 (582.67).

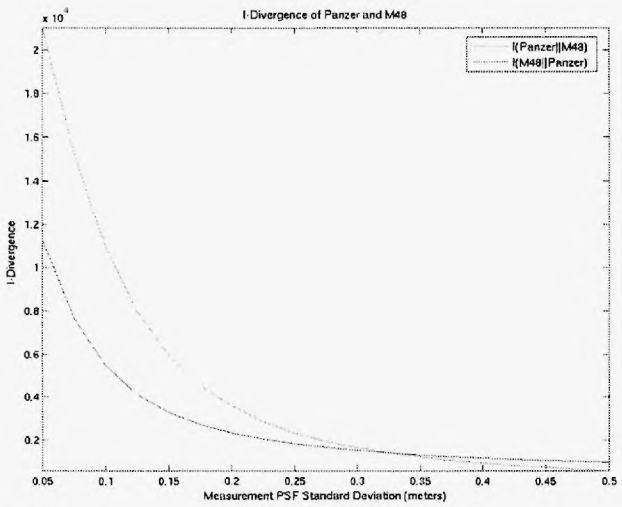


Figure 17: I-divergences between Panzer and M48 models.

A.D. Lanterman, "A Mathematical Theory of Automatic Target Recognition," *Workshop on Quantum Algorithms for Signal, Image, and Data Processing*, UCSD, La Jolla, CA, Dec. 7-9, 2004. Invited talk (trip paid for by UCSD).

A.D. Lanterman (with L.M. Ehrman), "Assessing the Performance of a Covert Automatic Target Recognition Algorithm," SPIE Defense and Security Symposium, Orlando, FL, March 28-April 1, 2005. (See paper reference above).

A.D. Lanterman (with J.H. Dixon and L. Fomundam), "Information-Theoretic Bounds on ATR Performance from Laser Radar Data," *AFOSR 2005 Program Review for Sensing, Imaging and Object Recognition*, NCSU, Raleigh, NC, May 25-27 (681.71). J.H. Dixon also attended (240.00)

A.D. Lanterman, "General Pattern Theory Applied to ATR," *ARO/AMRDEC Workshop on Information-Theoretic Image Processing*, Redstone Arsenal, Huntsville, AL, June 14-15, 2005. Invited talk (trip paid for by Army organizations).

A.D. Lanterman, "A Passive Radar Testbed at Georgia Tech," *4th Multinational Passive Covert Radar Conference*, Syracuse Univ. Hotel and Conference Center, Syracuse, NY, Oct. 5-7, 2005 (797.21).

M. Tobias (with A.D. Lanterman), "Using the Probability Hypothesis Density for Multitarget Tracking with Passive Radar," *4th Multinational Passive Covert Radar Conference*, Syracuse Univ. Hotel and Conference Center, Syracuse, NY, Oct. 5-7, 2005 (653.16).

5.2 Transition: LADAR Simulator

Greg Arnold of AFRL/SNAT became interested in our use of OpenGL in our custom-made laser radar and infrared simulation tools. Some of their previous efforts to cre-

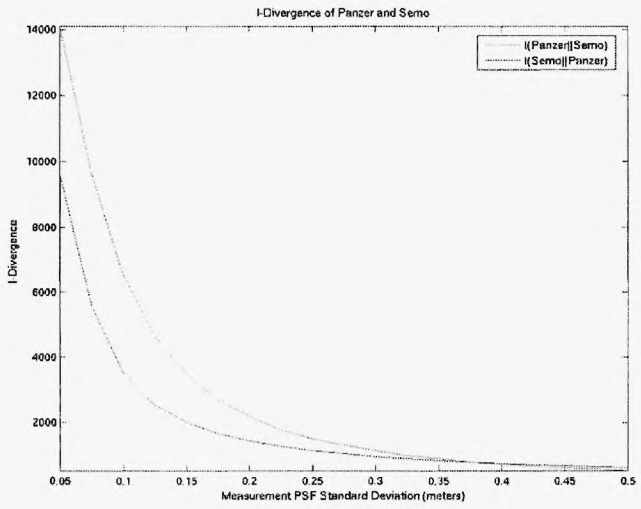


Figure 18: I-divergences between Panzer and Semovente models.

ate synthetic datasets for use by the ATR community involved POV-Ray and IRMA, each of which present various difficulties. We provided him with a documented copy of our simulation code, which he provided to Jeremy Olson and Kyle Erickson of AFRL/SNAA, who performed a detailed review and provided an extensive set of comments, which Jason Dixon used to prepare an updated version of the simulator. We went through several such review/update iterations.

The simulator was originally created to create scenes based on a "heterodyning" laser radars, such as those studied by Jeff Shapiro of MIT. Such radars have a characteristic ambiguity in range manifest in the banded structure of the background. We revised the code, particularly the noise generation code, to allow modeling of the "direct detection" radars studied by T.J. Klausutis of AFRL/MNGI and his colleagues.

Our resulting "LADAR Simulator" code, its associated user's manual, and cross-linked help pages for a small Application Programming Interface (API) that will let users develop their own MATLAB scripts using the simulator's scene generating capabilities is currently available via the web at `users.ece.gatech.edu/\~jdixon/ladar\_sim`. The user's manual is also attached to this report as Appendix A.

In an e-mail dated April 30, 2007, Greg Arnold informed us that his group completed a new Challenge Problem description and data set created using our LADAR Simulator, and that he was reviewing it and working on the "challenge experiments" to send for evaluation for public release.

6 Patent Disclosures

None.

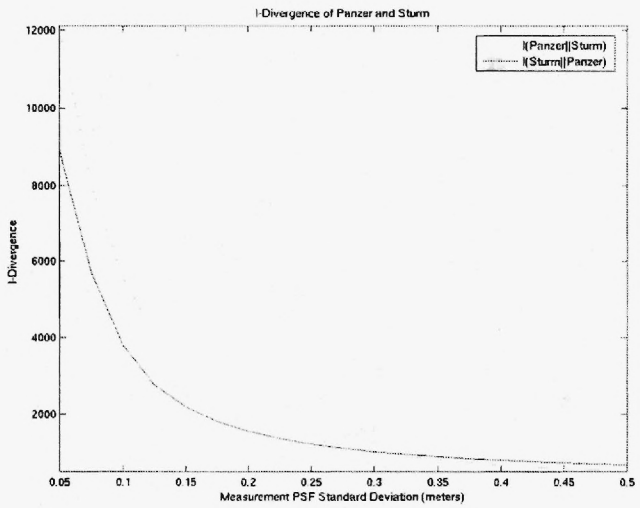


Figure 19: I-divergences between Panzer and Sturmgeschultz models.

7 Honors

A.D. Lanterman, Richard M. Bass/Eta Kappa Nu ECE Oustanding Junior Teacher Award (2006), as voted on by the senior class.

A.D. Lanterman, named the Demetrius T. Paris Junior Professor beginning in September 2004. This special three-year Chair position was founded to support the development of young faculty.

A.D. Lanterman, NIC Certificate of Excellence "for outstanding contributions to the National Intelligence Council and exceptional service to the Intelligence Community," April 2001.

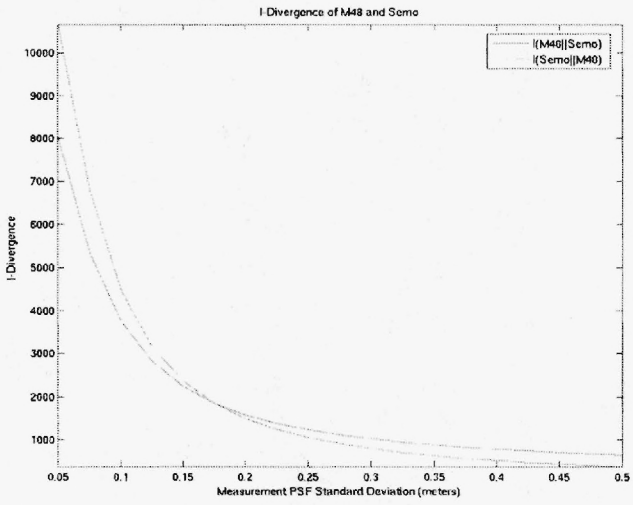


Figure 20: I-divergences between M48 and Semovente models.

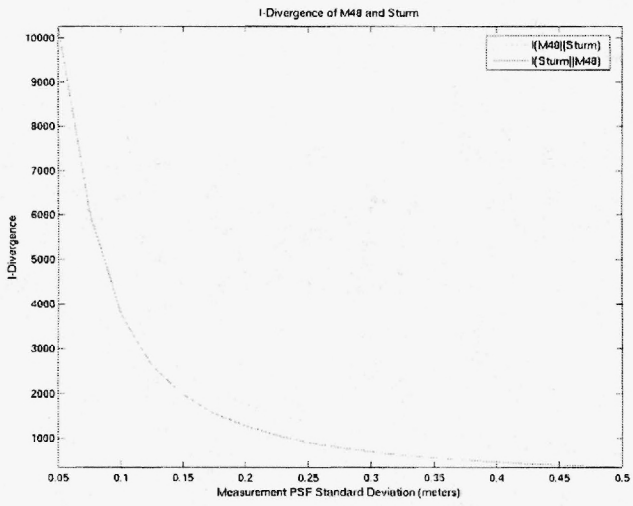


Figure 21: I-divergences between M48 and Sturmgeschultz models.

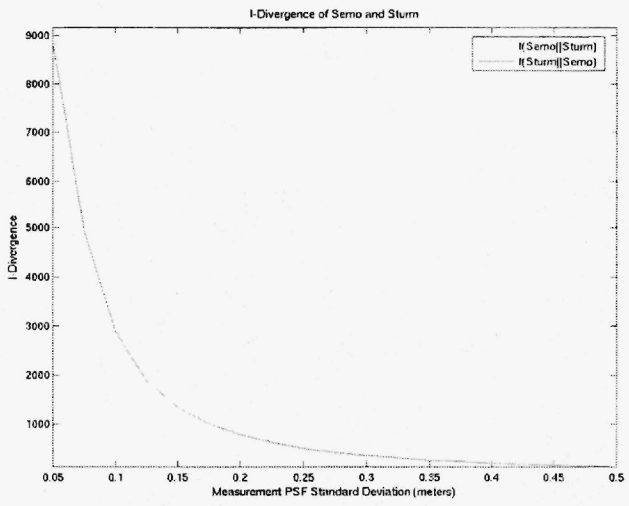


Figure 22: I-divergences between Semovente and Sturmgeschultz models.

Appendix A

J.H. Dixon, *LADAR Simulator User Manual.*

LADAR Simulator User Manual

Jason H. Dixon
Email: jdixon [at] ece [dot] gatech [dot] edu

Version 0.7

January 16, 2007

Contents

Abstract

This document describes a simulation tool used to created synthetic range imagery and three-dimensional point cloud datasets within MATLAB. Created scenes consist of multiple faceted models at user-specified coordinates, orientation angles, and sizes. The tool consists of a graphical user interface (GUI) and a simple library interface. The GUI aids scene creation by facilitating object placement, specifying the ground plane, defining the viewing perspective, and adjusting model parameters. The library interface allows the use of the scene generation code to write MATLAB scripts that create and analyze customized data sets. The tool currently supports faceted models in PRISM, 3D Studio, stereolithography, and Princeton Shape Benchmark object file formats.

Chapter 1

Getting Started

1.1 System Requirements

System requirements for running the LADAR Simulator are:

- MATLAB Version 7.0 or above with the LCC compiler
- Installed OpenGL libraries
- Operating System: Windows XP, Mac OS X[1]

There is no recommended requirement for RAM, CPU speed, or hard disk space, except for what is required for MATLAB. Please keep in mind that rendering three-dimensional graphics is a resource intensive process. The software has been tested on 2.0 GHz Pentium 4 system with 512MB of RAM, producing satisfactory results.

1.2 Installation

Follow these steps to set up the LADAR Simulator:

1. Extract the contents of the archive `ladar_simulator.zip`.
2. Move the folder named `ladar_simulator` to a desired location.
3. Start MATLAB and navigate to that same `ladar_simulator` directory.
4. The MATLAB compiler needs to be set to LCC. Run the command `mex -setup`. MATLAB will ask you if you want to locate installed compilers. Type `y` and press the `Enter` key. One of the resulting choices should begin with `Lcc C` followed by a version number and directory. Type in the number corresponding to that choice and press the `Enter` key. MATLAB will ask you to verify your choice.

[1] GUI layouts may not look right or be fully functional in MATLAB 7.0 for Mac. This issue was resolved with version 7.3 R2006b.

5. In the MATLAB prompt, run the command `install_ladar_sim`. This will compile the relevant C files and add the folder to the MATLAB path.

You are now ready to start using the LADAR Simulator. It can be run from any working directory you choose. To uninstall the LADAR Simulator, simply remove the directory from the MATLAB path and delete the folder.

If you do not wish to install the LADAR Simulator, you may choose to run all GUIs and programs from within the `ladar_simulator` folder itself, but you must first run the command `make_ladar_sim` to compile the necessary source files.

1.3 Files and Folders

All files are located in the folder `ladar_simulator`. This folder contains the MATLAB functions/scripts and C files for creating simulated range scenes and point clouds. Other files in the directory include

- `sample_models.ml` - model library of simple models

There are a number of sub folders containing example files:

- `config` - scene configurations
- `models` - simple faceted models
- `scenes` - range images and point clouds

The rest of the folders contain program-related libraries.

Chapter 2

Scene Creation Overview

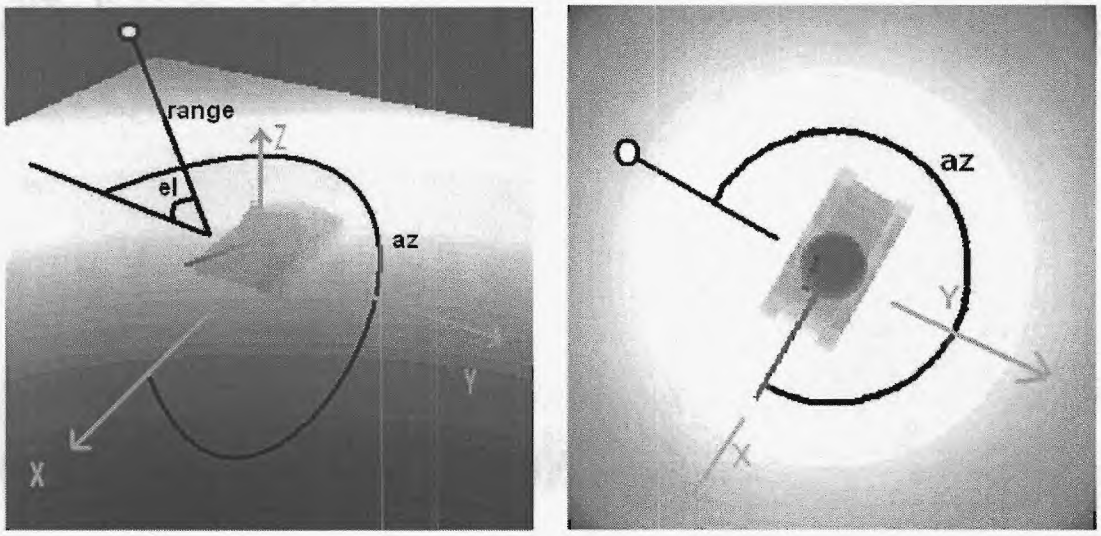
This chapter provides a high-level overview of the simulator and using it to define and render scenery.

2.1 Parts of a Scene

The LADAR Simulator tools allow users to create synthetic LADAR data sets (range images and point clouds) from predefined, three-dimensional scenes. The tool starts by setting up a three-dimensional world view of a particular scene in a space with axes x, y, and z. x and y represent coordinates along the ground, or any plane at some given height (as in the standard three-dimensional coordinate system used in computer graphics). The z-coordinate represents the height at particular x and y values. If we consider the ground, then all z values will be zero. A viewing sensor is usually positioned at some point above the $z = 0$ plane, pointing at some predefined point in the world. The position of the sensor may be defined in rectangular x, y, and z coordinates, or in terms of range, azimuth, and elevation spherical coordinates based on the "look at" point. A flat, rectangular ground plane of any desired size can be placed at the coordinates where $z = 0$ to give the appearance that objects in the scene are resting on a surface. The units of the world coordinates are not explicitly set and may represent anything. For example, let us say that you want to view a tank from 50 meters away, but the faceted model of the tank is defined in millimeters with dimensions $6367 \times 3122 \times 2943$. Specifying 50 as the range will result in an appearance that may be interpreted in two different ways: (1) the tank is 6367mm $\times$3122mm $\times$2943mm and the sensor is 55mm away, or (2) the tank is 6367m $\times$3122m $\times$2943m and the sensor is 50m away. Without scaling the values, the distance from the target and the dimensions of the tank are considered to be the same units. To have all objects in the scene drawn appropriately, the units of the tank must be converted to meters by scaling by 1/1000, or the units for the sensor's range from the target must be specified in millimeters.

2.2 Setting the View

The viewing area and resolution of the resulting imagery are set via the field of view and data dimension parameters. Field of view is defined in the horizontal and vertical directions, relative to the position of the sensor, in units of degrees. In essence, the sensor scans +fov/2 and -fov/2 in both directions. The number of pixels in the image is determined by the data dimension parameters (think of this parameter as the angular sampling that is occurring as the sensor scans over the field of view angles). Square images are created by setting the two field of view angles to the same value while rectangular images result from one field of view angle being larger than the other, regardless of the pixel dimensions. The sensor's position can be specified in world or spherical coordinates. To use world coordinates, simply set the $[x, y, z]$ parameter vector. For spherical, set the $[\rho, \theta, \phi]$ coordinates, representing range, azimuth, and elevation. An illustration of the view and coordinate system can be seen in Figure 2.1. Range is taken to be the distance between the sensor and the "look at" coordinate, also specified as a $[x, y, z]$ parameter vector. The minimum and maximum viewable ranges must also be set using the appropriate parameters. When viewing point clouds, the data dimension and field of view angles are used to defined the scene from which the points will be extracted.



(a) (b)

Figure 2.1: A Standard (a) and overhead (b) view of a scene's layout. The sensor location is represented by the white circle.

2.3 Adding Targets

Scenes may contain any number of objects in a variety of positions and orientations. These objects may be referred to as "targets" throughout this document. Target geometries are

obtained by reading in CAD model descriptions. The CAD model formats that this software supports are PRISM, Stereolithography, 3D Studio, and Princeton Shape Benchmark. Targets are placed at specified (x, y, z) coordinates and rotated by some angle around the three axes. Targets may be rescaled if the units used to define the target's CAD model are undesirable.

Chapter 3

Graphical User Interfaces

This chapter reviews the different graphical user interfaces (GUIs) used in the LADAR Simulator.

3.1 Main LADAR Simulator GUI

To run the LADAR Simulator GUI, type `ladar_sim` in the MATLAB prompt. The GUI will prompt the user for a configuration file to load. This file contains information for setting up a scene. An example configuration file can be found in `ladar_simulator/config/sample.config`. If you wish to load default configuration settings, just press the `Cancel` button in the dialog box.

Next, another dialog box will appear. This box asks the user to load a model library file. This file contains structural information about the models referenced in the configuration file. If a configuration file was selected, a predetermined model filename will be chosen and the user must locate that file. If no configuration file was selected, the user is free to specify any available model library file. The default model library file is `ladar_simulator/sample.ml`. If no model libraries are available, the user may create one by pressing the `Cancel` button and using the Model Library Creation GUI. For instructions on how to use this GUI, see Section 3.3. For the LADAR Simulator GUI to run, a model library must be selected or created.

Once the model library has been chosen, the LADAR Simulator GUI will appear (see Figure 3.1). The GUI contains all components necessary to create a scene. The axes on the right side of the window displays the rendered configuration or point cloud. The type of scene is selected from three possible choices using the pop-up menu below the scene axes: `Full Range`, `Point Cloud`, `Depth Buffer`. For a detailed description of these scene types, see Section 5.3. By default, the color mapping used for two-dimensional data sets is determined automatically by the minimum and maximum values in the scene. This can be overridden by marking the `Colormap` check box and selecting the minimum and maximum values values for color map scaling. The user can add or remove a ground plane to the scene by selecting the `Ground Plane` check box. The dimensions of the ground plane can be defined by pressing

the adjacent `Options` button. For details about setting up a ground plane, see section 3.5. Gaussian distributed noise of a chosen variance may be added to the scene by selecting the `Noise` check box. The noise is centered on the range values (or coordinates in the case of point cloud data).

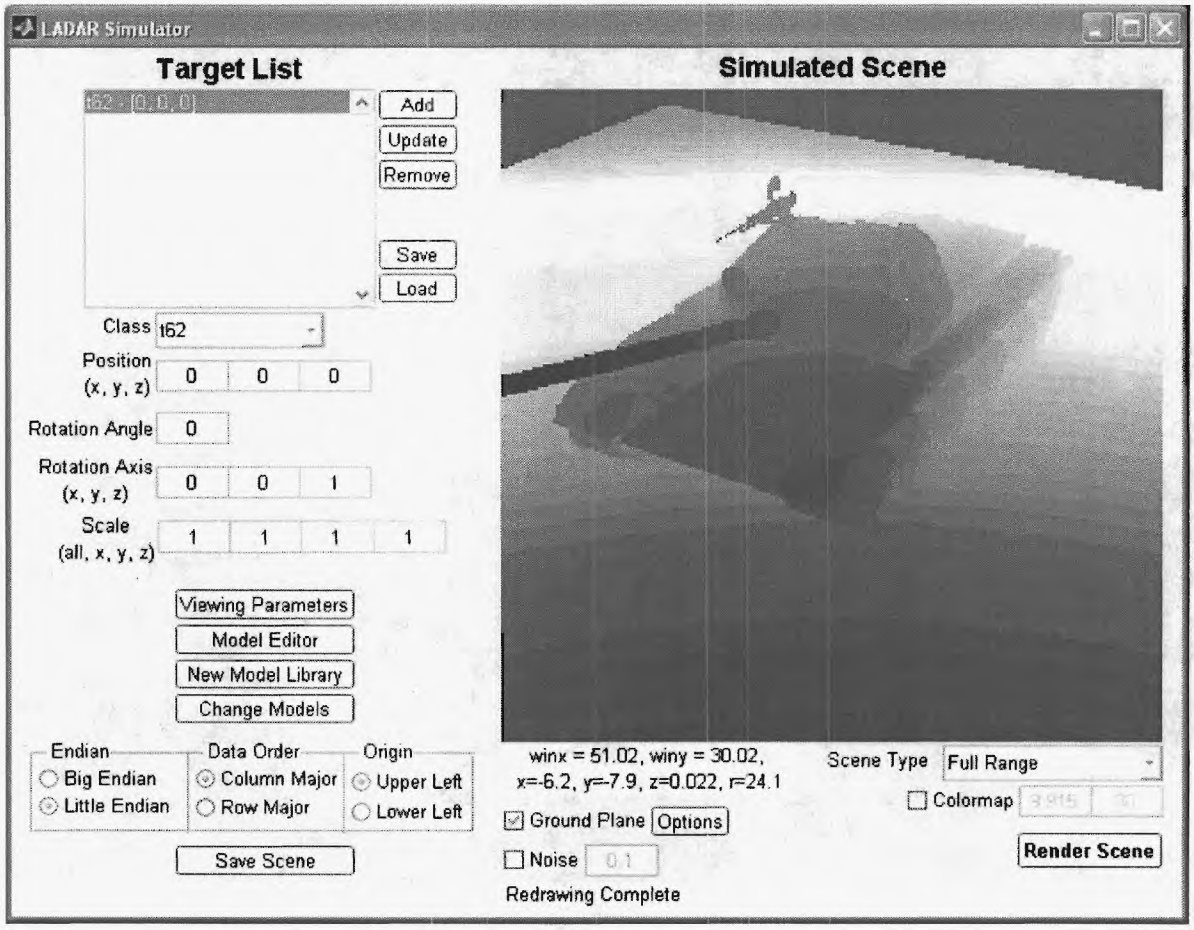


Figure 3.1: Main LADAR Simulator GUI window.

The *Target List* listbox shows targets that have been added to the current configuration. The text fields below the list box are used to set target parameters. Targets may be added, removed, or updated by pressing the corresponding buttons next to the `Target List` list box. Targets may also be added with the mouse by left-clicking on the desired location in the scene. When you select an item in the listbox, the corresponding target parameters will appear in the text fields. Users may also update existing targets by pressing the `Enter` key after changing one of the text field values. Scene configurations may be loaded from configuration files by pressing the `Load` button. The `Save` button will save a configuration to a file. The adjustable target parameters are as follows:

- Position: the location of the target in the scene. x and y are ground coordinates, while z is the coordinate above or below the ground plane. In most situations, z will be set to zero.
- Rotation Angle: an angle rotation in degrees starting from the positive x-axis and moving counter clockwise.
- Rotation Axis: axis about which to perform rotations. To perform rotations about z-axis, this should be set to $[0, 0, 1]$.
- Scale: stretches or shrinks a target by multiplying the scale by the coordinates used to define the target.

From within the main LADAR Simulator GUI, other GUIs may be launched. To change the viewing perspective and associated parameters, press the `View Settings` button (see Section 3.2). To adjust CAD model parameters, press the `Model Editor` button (see Section 3.4). To create new model libraries, press the `New Model Library` button (see Section 3.3). To change the current model library, press the `Change Models` button.

To save range images or point clouds, users can use the corresponding button and radio toggles. When saving data sets, it's best to remember the format used because it will be necessary information for correctly reloading datasets. By default, data is saved in the native machine format, column major, with the upper left corner of the image set as the origin.

3.2 View Settings GUI

The View Settings GUI facilitates the adjustment of scene viewing parameters (see Figure 3.2). The text fields of the GUI are as follows:

- Field of View: two fields representing the vertical and horizontal scanning angles used to determine the viewing area. The angles are interpreted as + or - fov/2 from the current "look at" position in either direction.
- Position (cartesian): $[x, y, z]$ coordinates representing the sensor location. When the cartesian coordinates are changed, the spherical coordinates are adjusted automatically.
- Position (spherical): $[\rho, \theta, \phi]$ coordinates representing the range, azimuth, and elevation from the "look at" point in the scene. When the spherical coordinates are changed, the cartesian coordinates are adjusted automatically.
- Look At: cartesian coordinates representing the point to which the sensor is pointing. When the "look at" point is changed, the spherical coordinates are adjusted automatically. It is assumed that changing the "look at" point does not reflect a change to the sensor position.

- Min/Max Range: two fields representing the placement of the clipping planes relative to the sensor location. Targets or part of the ground plane placed beyond these range limits will not appear in the rendered scene. Pixel values in range images that represent scene elements beyond the range limits will be clipped to the minimum and maximum range values.

- Dimensions: two fields representing the vertical and horizontal (number of rows versus number of columns) pixels in range images. For point clouds, these fields represent the number of samples used to extract point cloud information (in the event that all samples are valid points, then the product of these two fields is the number of points in the point cloud). When combined with the field of view angles, these fields can be thought of as the angular scanning sampling in the vertical and horizontal directions.

- Up Vector: supplies the rendering system with a vector orientation for up. In most cases, these may be set to $[0, 0, 1]$, representing the positive z-axis as the up direction. If the camera is pointed straight down, then a new up direction must be chosen so that the vector has a projection along the xy-plane.

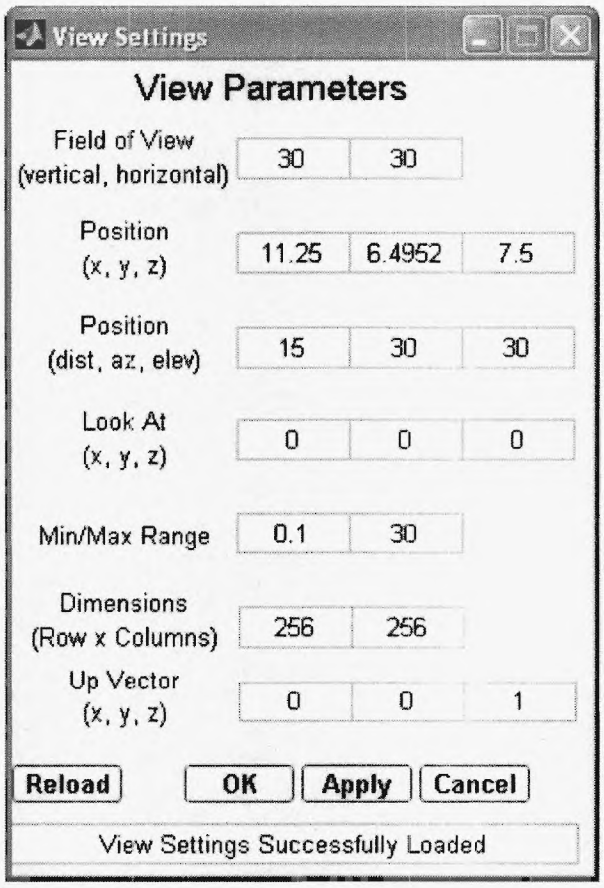


Figure 3.2: View settings GUI window.

When changing fields, the GUI will perform a general validity check for the most recently used parameter and revert to the original value if the current value is found to be invalid. Pressing the `Reload` button will reset to the GUI to its original state after it was first launched. Pressing the `OK` button will save the view settings, close the View Settings GUI, and redraw the scene using the new settings. Pressing the `Apply` button will save the current view settings and redraw the scene. Pressing the `Cancel` button will close the View Settings GUI and discard all changes since the last time the `Apply` button was pressed. Pressing the `Enter/Return` key after editing a text field simulates pressing the `Apply` button.

3.3 New Model Library GUI

The LADAR Simulator GUI allows users to create their own model library files consisting of references to CAD models in supported model file formats and a set of adjustable parameters. When the "New Model Library" button in LADAR Simulator GUI window is pressed, a dialog box will appear that allows the user to select the name and path of the model library

file to be created (see Figure 3.3). The file must be saved to a directory in, or above, where the CAD model files are stored. Once the file has been specified, the New Model Library GUI will appear, as shown in Figure 3.4.

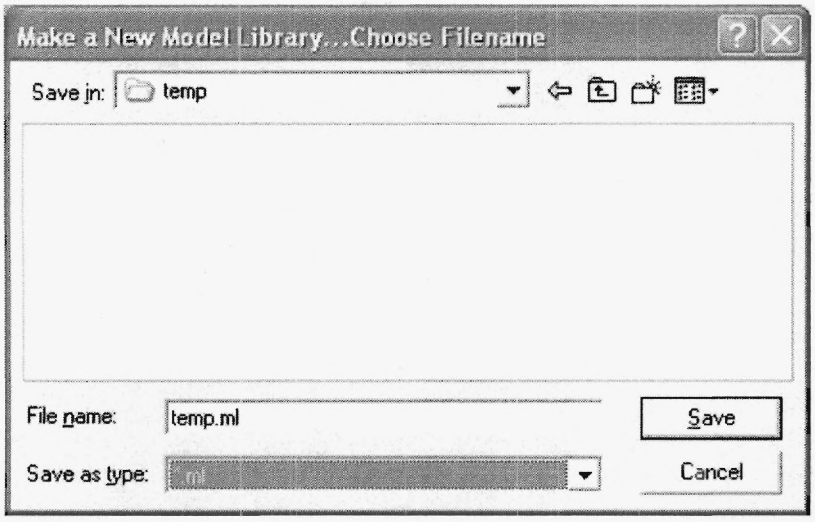


Figure 3.3: Choose the filename and path for the model library file.

The New Model Library GUI is an interface used to create a model library file with chosen model CAD files (see Figure 3.4).

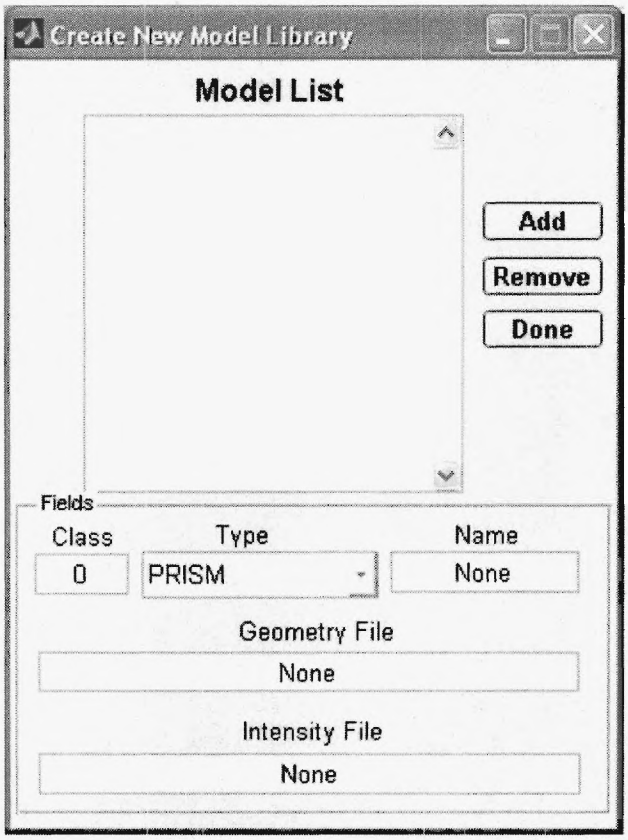


Figure 3.4: Empty model library creation GUI.

The listboxes and fields are defined as follows:

- Model List: listbox that displays the model files currently in the library. When a model is selected, the corresponding parameters will be shown in the GUI text fields.
- Class: a unique nonnegative integer used to identify a model in the library.
- Type: a pop-up menu that allows you to choose among CAD file types. These include STL, PRISM, 3DS, and OFF.
- Name: a text string identifier for the model
- Geometry File: path and location of the geometry CAD model file relative to the location of the model library file.
- Intensity File: the same file and path as in the Geometry File field, unless working with PRISM models. In that case, use the radiance filename.

For more information about the meaning of these fields, see Section 5.2.3.

To add model files to the library, press the Add button. A file dialog like the one shown in Figure 3.5 will appear. Navigate to the subfolder with the geometry files and select the one(s) you wish to add to the model library. Once you click Open, the Model List listbox will populate with the chosen models and you can see the corresponding text fields by clicking on a model in the list (see Figure 3.6). By default, the classes are numbered sequentially in the same order in which the files were selected. The types are determined by the extension of the geometry file. The names are determined by the filename. Any of the fields may be changed from the their default values. The Remove button will remove the currently selected model from the library. When finished, press the Done button and the model library creation GUI will close, the new model library will be loaded, and you will be returned to the main LADAR Simulator GUI.

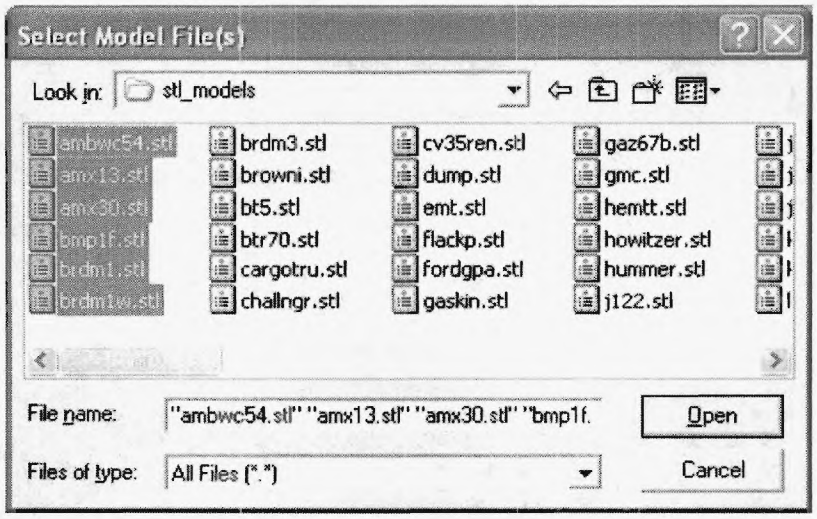


Figure 3.5: Dialog box for selecting CAD model files to add to the model library.

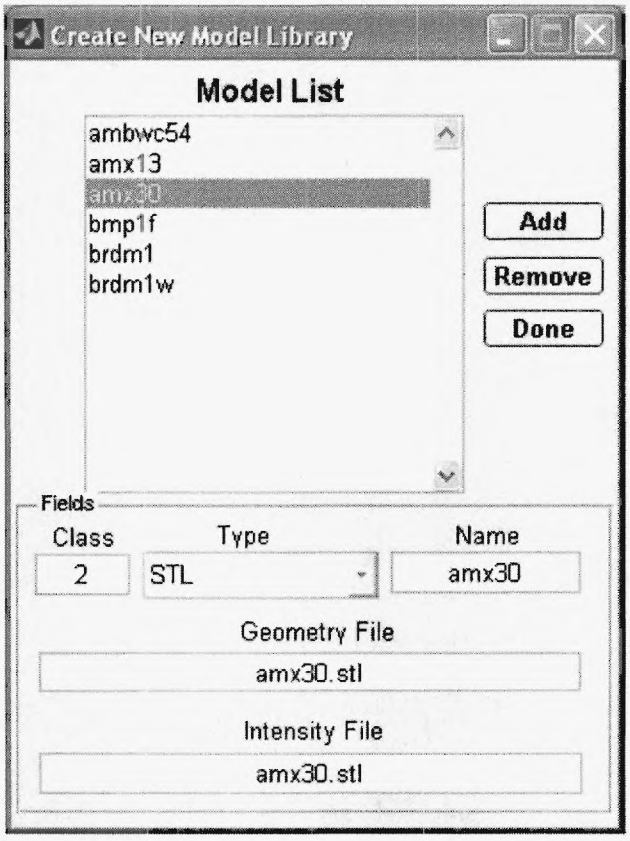


Figure 3.6: Model library creation GUI after adding CAD model files.

Please note: When a model file is read, all geometry files referenced therein will be loaded into main memory. If you have too many large geometry files referenced in a model library, your computer may not be capable of loading the model library. Please consider the memory capabilities of your computer, and the size of your model geometry files, when creating model libraries.

3.4 Model Editor GUI

Model geometry files are created in many different ways. The units used to define the models and the orientation in which a model rests can vary greatly from file to file. In addition to organizing a collection of model files, model libraries hold information about resizing, rescaling, and reorienting the models themselves. These adjustments allow you to redefine the default units and orientation of a model without permanently altering the geometry file. The Model Editor GUI can be launched by pressing the `Model Editor` button in the main LADAR Simulator GUI (see Figure 3.7).

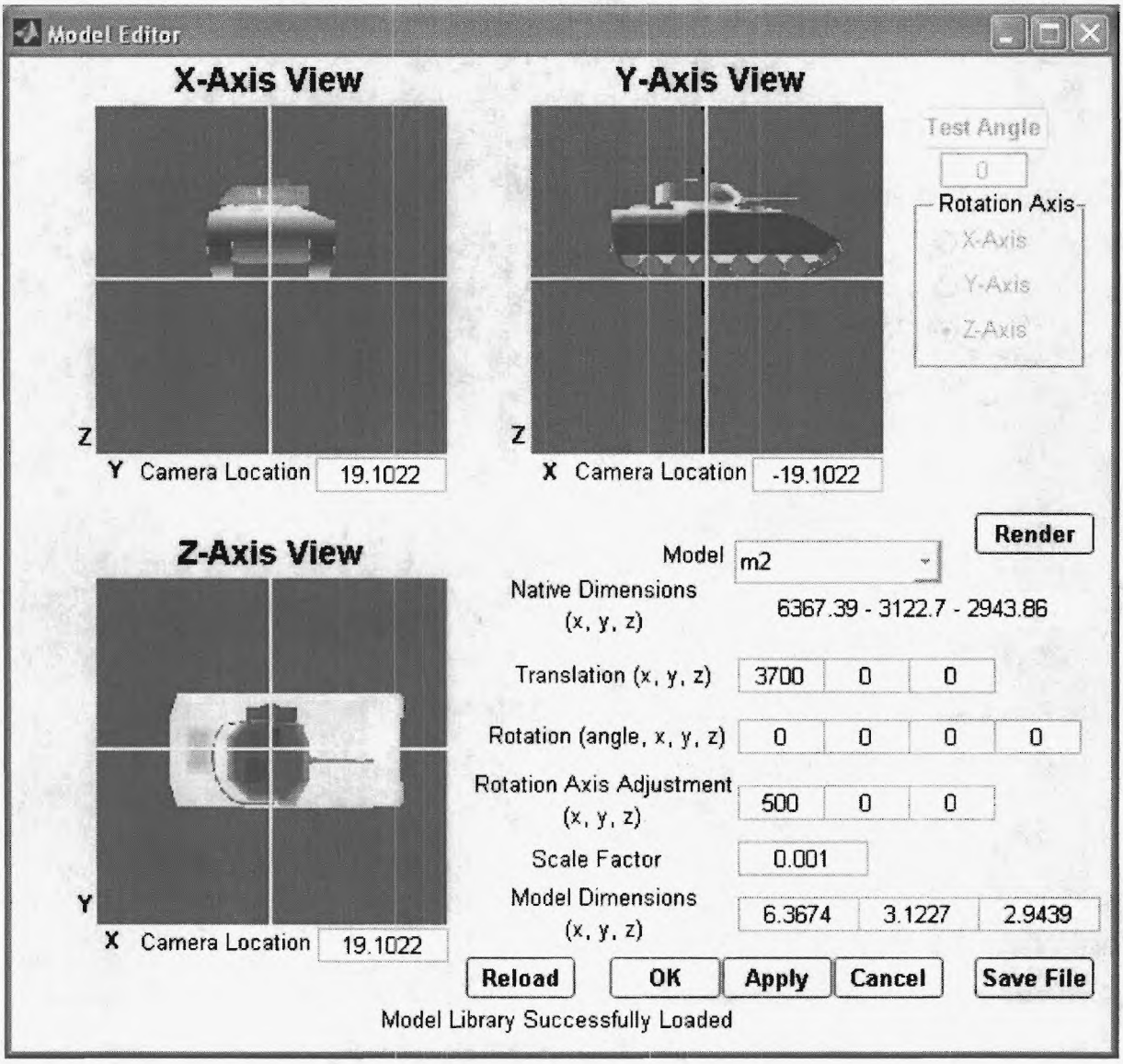
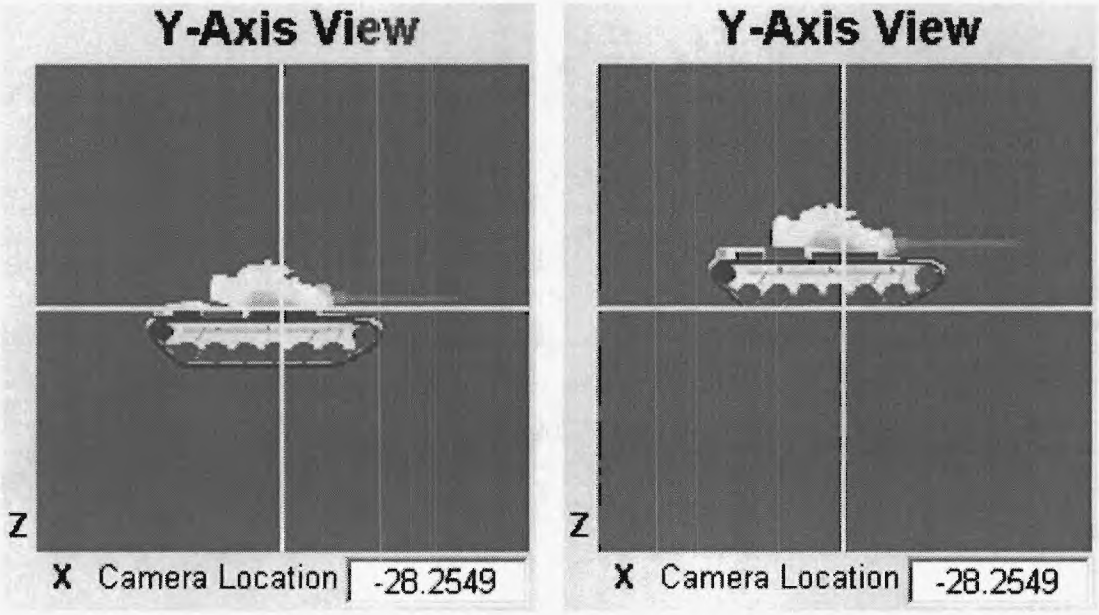


Figure 3.7: GUI for changing model library parameters.

The GUI has three windows, where each window faces the origin from one of the three coordinate axes. The GUI attempts to automatically adjust the viewing distance in each window so that the model is completely visible. If a different view is desired, you can change the value underneath the window. This value represents the distance between the camera and the origin along the specified axis. To "zoom in" on the object in the window, simply decrease the magnitude of this value. To "zoom out," increase the magnitude of this value.

The `Model` pop-up menu allows you to choose the model to be adjusted. The `Translation` text fields are used to define a default movement to occur in each direction before an object

is placed in a scene. Typically, model files are defined in such a way that the origin is considered to be one of the bottom corners of the object. It may be convenient to place the origin on the bottom center of the model so that if a target is placed at $[0, 0, 0]$, then the target will straddle the center point. Using the *same units as the geometry file*, you can specify desired translations along the x, y, or z axes. An example is shown in Figure 3.8.

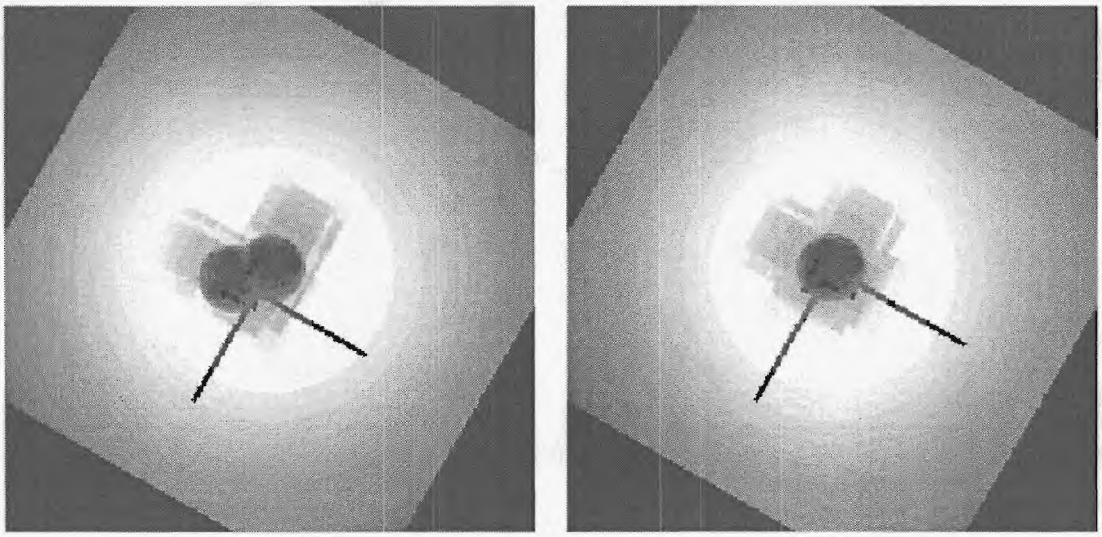


(a) (b)

Figure 3.8: Images showing a tank before (a) and after (b) a translation adjustment along the $z-$axis. After the adjustment, tanks placed in a scene will not be halfway in the ground plane.

The `Rotation` text fields are used to define a default rotation to occur before an object is placed in a scene. Sometimes model files are defined in such a way that leaves them in an unnatural orientation when used in a scene. For example, a car model may be sitting on its bumper by default. These text fields allow you to define a rotation along any of the axes that will occur before a model is placed. In the case of the car, you can define a rotation that places the car on its wheels. The rotations are specified in the angle-axis notation, where you specify a rotation angle and the vector form of an axis about which to rotate.

The `Rotation Axis Adjustment` text fields allow you to adjust the position of a model before a rotation occurs. By default, rotations occur on an axis that in the center of the model's bounding box. For symmetric models, this works out well. When there are features that take away from this symmetry, such as an extra long gun extending from a tank, it may be desirable to adjust the axis about which the rotation occurs. In the case of the long gun, the center of the bounding box may be closer to the front of the tank and a rotation would appear awkward. An example of this effect is shown in Figure 3.9.



(a) (b)

Figure 3.9: Images showing how adjusting the rotation axis affects the final rotation. Both images show two T62 tanks, one at the default orientation and another at the same location but rotated by 90 deg. There is no rotation axis adjustment in (a) so the rotation axis is further from the center of the tank body. This gives the rotated tank the appearance that it was displaced slightly. In image (b), the rotation axis is moved so that it is in the center of the tank body. This results in a rotation as we would intuitively think about one.

The `Native Dimensions` field shows the extent of the current model along the $x-y-z$ axes. Think of this as the dimensions of the bounding box along each of these axes. The values in this field are determined by the native geometries of the model file, meaning that they are the same as in defined in the CAD model geometry file. The `Model Dimensions` text field shows how the extents change after applying the value in the `Scale Factor` field. The scaling can also be adjusted by entering a new value directly in the x, y, or z `Model Dimensions` field. For example, a model's native dimensions are defined in a such a way that you have a vehicle that is 5000 units long in the x-direction, and you know that the vehicle is supposed to me 4 meters long in that direction, then you can type 4 into the x field. All other `Model Dimensions` will adjust accordingly, as will the scale factor. The camera positions in the three windows will also be adjusted to provide the same view of the object after they have been resized. Note that when adjusting the `Translation` or `Rotation Axis Adjustment` fields, the units are in the native model units and not the scaled ones.

When using the Model Editor GUI, some trial and error is usually involved in order to obtain desired model adjustments.

3.5 Ground Plane GUI

The Ground Plane GUI provides users access to the geometric description of the ground plane (see Figure 3.10). The ground plane is a flat, rectangular surface. A corner is defined by setting the $[x, y, z]$ coordinates of the "Origin" field. The lengths in the x and y directions are specified using the next two text fields. If negative values are supplied in these fields, then the ground plane extends along the axes in the negative direction. An arbitrary intensity value is specified in the last text field. The value itself is arbitrary and may be set to any positive integer.

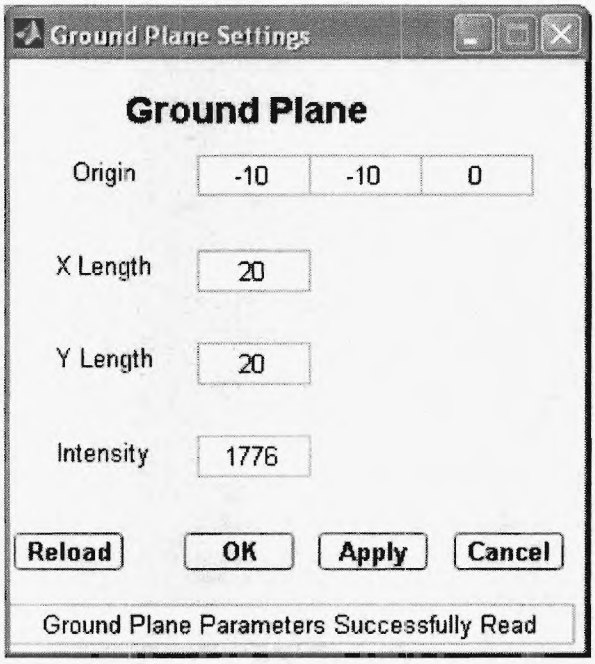


Figure 3.10: GUI for changing ground plane settings.

When changing fields, the GUI will perform a general validity check for the most recently used parameter and revert to the original value if the current value is found to be invalid. Pressing the `Reload` button will reset to the GUI to its original state after it was first launched. Pressing the `OK` button will save the ground plane settings, close the Ground Plane GUI, and redraw the scene using the new ground plane. Pressing the `Apply` button will save the current ground plane settings and redraw the scene. Pressing the `Cancel` button will close the Ground Plane GUI and discard all changes since the last time the `Apply` button was pressed. Pressing the `Enter/Return` key after editing a text field simulates pressing the `Apply` button.

Chapter 4

Library Interface

The functions used to read in models, create/manipulate scene configurations, and draw/display scenes are all available to call directly from a user-defined MATLAB script or function. This capability was included to allow users to run their own simulations using synthetic data generated on-the-fly.

Some of the functions and their descriptions are described here:

- Loading and Saving Files
  - `load_configuration` - Load configuration file into structures
  - `load_data` - Retrieve image or point cloud from data file
  - `load_model_library` - Read model library file into a structure
  - `save_configuration` - Save configuration to a text files
  - `save_data` - Save a range image or point cloud to a data file
  - `save_model_library` - Save model library structure to a text file
- Manipulating Configuration Structures
  - `add_target` - Add target with given parameters to configuration
  - `new_config` - Create an empty configuration structure
  - `remove_target` - Removes a target from a configuration
  - `set_ground_param` - Update the ground plane in a configuration
  - `set_target_param` - Update a single target in a configuration
- Manipulating View Structures
  - `new_view_settings` - Create a view settings structure
  - `set_view_param` - Update a view setting parameter
- Working with Scene Data

  - `add_noise` - Adds noise to range imagery or point cloud data
  - `display_data` - Display a scene in a given figure or axes

- Scene Creation
  - `render_scene` - Create a range image or point cloud data set

Customized simulations typically start out with loading a model library and a configuration file (use `load_model_library` and `load_configuration`). If no configuration file is available, then a configuration is created from scratch with the `new_config` function with some ground plane information. Targets are added with the `add_target` function. The view is created and modified with the `new_view_settings` and `set_view_param` functions respectively, and so on.

The `ladar_simulator` folder contains a subfolder called `scripts` with an example simulation file named `make_multiview_ptc.m`. It creates a single point cloud from multiple views of the same scene, consisting of an object positioned in the middle of a ground plane. The sensor takes frames at 120 deg increments and merges the point clouds from the individual frames into a single point cloud. It then saves the point cloud and configuration structure to files using the `save_data` and `save_configuration` functions, respectively. Each range image is displayed along the way, as was well as the final point cloud, using the `display_data` function. This script may be used as a guide for creating other customized simulations.

Chapter 5

File and Structure Formats

Although the GUIs and library interfaces should be sufficient for most users, it may be desirable for users to directly manipulate the configuration files, model libraries, or the various structures. This chapter includes descriptions of these formats. If the tools provided for manipulating the structures and files are sufficient for your needs, then this chapter may be skipped.

When manipulating the structures directly, it is important to remember data types. By default, MATLAB stores everything as a double precision floating point value. Since these structures are passed to C functions that utilize external libraries, many of the fields in these structures are not double precision floating point. Therefore, it is important to use the appropriate data type when assigning values to the structure fields referenced in this chapter.

5.1 Configurations

The configuration file and structure format includes fields for organizing different components of a scene. These components include target parameters, view settings, and the ground plane description. All arrays are stored row-wise.

5.1.1 Configuration Structure

The configuration structure description is shown in Table 5.1. It holds information on the targets and the ground plane in the scene.

5.1.2 Target Structure

The target structure description is shown in Table 5.2. The *class* field holds a nonnegative integer that identifies a unique model in the corresponding model library. The *position* field represents the target's $[x, y, z]$ coordinate position. The *orientation* is stored in angle-axis notation, so that rotations occur by rotating the target θ degrees along the $[x, y, z]$ axis. In

Table 5.1: Configuration structure fields.

Field Name	Data Type	Description
numTargets	uint32 scalar	Number of targets in the scene
targets	structure array	Array of target structures
useGround	int32 scalar	-1 = Do not include a ground plane 0 = Use default ground plane 1 = Use customized ground plane defined in the configuration structure
groundIntensity	uint32 scalar	ground intensity (any nonnegative integer)
groundOrigin	double array	Three-element $[x, y, z]$ array representing the origin of the ground plane
groundXLength	double scalar	Length of the ground plane in the positive-x direction
groundYLength	double scalar	Length of the ground plane in the positive-y direction

most cases, rotations will occur about the z-axis, so the $[x, y, z]$ portion of this field should be set to $[0, 0, 1]$. In the future, more intuitive rotations, like Euler angles, will be supported. The *scale* field scales the dimensions of the current target by a total value of s. It then scales the corresponding dimensions (as defined in the CAD model) by an additional x, y, or z in each of those directions. By default, this array should be set to $[1, 1, 1, 1]$ for no additional scaling. The *visible* field is either set to zero or one and controls the visibility of the target in the configuration (for visible targets, leave this set to one). The library function that manipulates the target structure is `set_target_param`.

Table 5.2: Target structure fields.

Field Name	Data Type	Description
class	uint32 scalar	Unique target class ID
position	single array	Three-element array of target $[x, y, z]$ coordinates
orientation	single array	Four-element array of target $[\theta, x, y, z]$ orientation
scale	single array	Four-element array of target $[s, x, y, z]$ scale
visible	uint32 scalar	0 or 1 specifying whether to render a target

5.1.3 View Settings Structure

The view settings are stored in a separate structure with parameters defined in Table 5.3. The library functions that manipulate the view structure are `new_view_settings` and `set_view_param`.

Table 5.3: View settings structure fields.

Field Name	Data Type	Description
vfov	double scalar	Vertical field of view
hfov	double scalar	Horizontal field of view
position	double array	Three-element array of sensor $[x, y, z]$ coordinates
lookAt	double array	Three-element array of "look at" $[x, y, z]$ coordinates
upv	double scalar	Three-element array representing the up direction
clipping	double array	Two-element array consisting of the minimum and maximum ranges
dataDim	uint32 array	Row x Column dimension of the result image in pixels, (the vertical and horizontal sampling)

5.1.4 Configuration File Format

Figure 5.1 shows the structure of a configuration file. Configuration files can be loaded and saved with the `load_configuration` and `save_configuration` functions, respectively.

5.2 Model Libraries

Model libraries are a collection of model geometries and associated parameters for rendering. It is important to note that it is best not to have model libraries that contain too many highly detailed models because all model information is stored in main memory. If a user were to create a model library with 20 models and each model file was larger than 20 megabytes when loaded, then MATLAB would need over 400 megabytes of memory to store the model library (at bare minimum). Keep your system's memory requirements in mind when creating model libraries.

5.2.1 Model Library Structure

Model library structures never need to be manipulated directly, so this section may be skipped if desired. The Model Editor and New Model Library GUIs are currently the only means by which model libraries may be changed.

Model library structures contain two fields: `numModels` and `models`. `numModels` is a field of type uin32 that stores the number of models in the library structure. `models` is an array of model structures, defined in Section 5.2.2.

5.2.2 Model Structure

Model structures never need to be manipulated directly, so this section may be skipped if desired.

```
model_library <filename>

v_field_of_view <degrees>
h_field_of_view <degrees>
camera_position <x position> <y position> <z position>
object_position <x position> <y position> <z position>
up_vector <x position> <y position> <z position>
clip_distance <min range> <max range>
data_dims <vertical pixels> <horizontal pixels>

number_of_targets <nonnegative integer>

target 1
class_of_target <nonnegative integer ID>
position <x position> <y position> <z position>
orientation <degrees> <x axis> <y axis> <z axis>
scale <global value> <x position> <y position> <z position>

...

target <final target number>
class_of_target <nonnegative integer ID>
position <x position> <y position> <z position>
orientation <degrees> <x axis> <y axis> <z axis>
scale <global value> <x position> <y position> <z position>

use_ground <-1, 0 or 1>
intensity <nonnegative integer>
origin <x position> <y position> <z position>
x_length <value>
y_length <value>
```

Figure 5.1: Configuration file format.

The fields of a model structure are defined in Table 5.4. Section 5.2.3 goes into more detail about the purpose of some of these fields. In the case of `gFile` and `rFile`, these will most likely be the same unless the CAD model is defined in the PRISM format. For CAD models that have no intensity information, the relevant fields are populated with arbitrary values when the models are first read from the files.

Table 5.4: Model structure fields and descriptions.

Field Name	Data Type	Description
class	uint32 scalar	numerical identifier for the model
type	char array	text description of the model type
name	char array	text identifier for the model
translate	single array	three-element (x, y, z) translation from the model's origin
rotate	single array	four-element (θ, x, y, z) default model rotation
rotateAdjust	single array	three-element (x, y, z) translation of the model rotation axis
scale	single scalar	global scaling of model units
gFile	char array	file and path of geometry file
rFile	char array	file and path of radiance file (if available), or `gFile`
vertexTable	single matrix	$N \times 3$ matrix of model vertices
vertexTableLength	uint32	N as used in `vertexTable`
facetIndexLists	cell array	$M \times 1$ array of uint32 arrays of indices into `vertexTable`
facetIndexListSizes	uint32 array	$M \times 1$ array containing the sizes of each array in `facetIndexLists`
numberOfFacets	uint32 scalar	the M as referenced above (number of facets in model)
intensityRegion	uint32 array	the intensity index of each facet
maxVertex	single array	three-element (x, y, z) for the largest bounding box coordinate
minVertex	single array	three-element (x, y, z) for the smallest bounding box coordinate
intensityList	uint32 array	$K \times 1$ array of intensity (or arbitrary) values
intensityListLength	uint32 scalar	K as referenced above (number of intensities)

5.2.3 Model Library File Format

The model library file format is designed to keep a record of CAD models common to a certain scene and the adjustable parameters for each of those models. To load or save model library files, use the `load_model_library` or `save_model_library` functions, respectively. The first line in the file contains the term `number_of_models` followed by a single space and an integer

representing the number of models in the file. Each model is listed on consecutive lines and the model parameters on a single line are separated by a single space. The parameters are as follows:

1. A unique nonnegative integer model identifier (in the configuration structure/file, this integer is the same as the class).

2. Type of CAD model (`STL` for ASCII or binary stereolithography files, `3DS` for binary 3D studio files, `PRISM` for files in the Prism file format from Thermoanalytics, and `OFF` for Princeton shape benchmark files).

3. Relative path (starting from the location of the model library file) and filename of the CAD model

4. A repeat the previous path and filename for non-Prism files. For Prism files, this field contains the corresponding radiance file.

5. The number `0` (not used, but left in for code legacy purposes).

6. A unique text string ID for the model. This will be displayed in the pop-up menus of the GUIs that allow that users to switch model types when creating a scene.

7. An array of three numbers in the form `[x,y,z]` (including the brackets and commas). This represents a coordinate translation from the model's origin as defined in the units that the model was created in. By default, this field is set to `[0,0,0]`. When changed, this field allows users to define another point in the model space as the origin. As an example, imagine rendering a model at the point $[0, 0, 0]$ in the scene coordinates. In many instances, this has the effect of rendering a lower corner of the model at that point. If the user would like for all models to be centered at the point of placement, then one could adjust the x and y parameters of this vector to first translate the model by these coordinates.

8. An array of four numbers in the form `[`θ`,x,y,z]` (including the brackets and commas). Similar to the previous field, this one represents a default rotation to take place for rendering a model. By default, this can be set to `[0,0,1,0]`.

9. An array of three numbers in the form `[x,y,z]`, including the brackets and commas. This represents a translation to take place before a rotation occurs. By default, this may be set to `[0,0,0]`.

10. A scalar value that represents a scaling of the coordinates in the model file. By default, this may be set to `1`.

For an example model library file, see Figure 5.2.

```
number_of_models 3
0 OFF mod/shape1.off mod/shape1.off 0 shape1 [0,0,0] [0,0,0,0] [0,0,0] 1
1 OFF mod/shape2.off mod/shape2.off 0 shape2 [0,0,0] [0,0,0,0] [0,0,0] 1
2 OFF mod/shape3.off mod/shape3.off 0 shape3 [0,0,0] [0,0,0,0] [0,0,0] 1
```

Figure 5.2: Example of a model library file.

5.3 Data Sets

The LADAR Simulator can create data sets in two forms: range images and point clouds. Data sets can be saved or loaded with the functions `save_data` and `load_data`, respectively. Data is displayed in MATLAB with the function `display_data`.

5.3.1 Range Imagery

Range images are rectangular grids where each element represents the range from the sensor to world coordinate captured by that element. In MATLAB, these images are stored in matrix form. See Figure 5.3 for a typical range image.

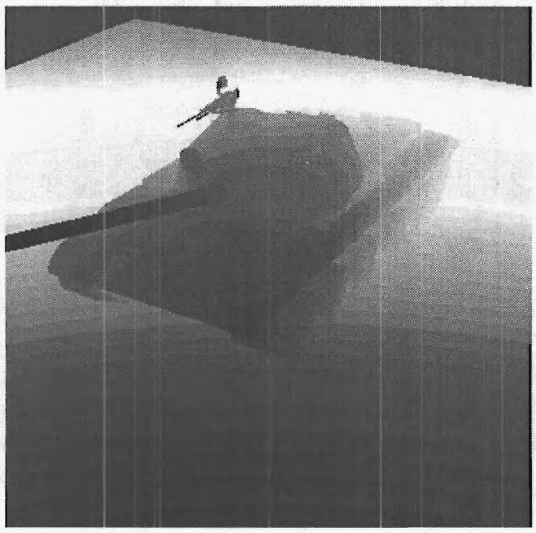


Figure 5.3: A typical range image.

5.3.2 Point Clouds

Point clouds are created by taking the two-dimensional points in a range image and converting them to their corresponding three-dimensional coordinates. The resulting point cloud consisting of L points is constructed as a $3 \times L$ matrix. If an image is rendered with a $N \times M$ data dimension, then there will be at most NM coordinates in the resulting point

cloud. The number of coordinates L is less than NM when there is sky present in the scene, when a ground plane is not rendered, or when pixels are set at the minimum or maximum range values. A typical point cloud is shown in Figure 5.4. It is worth noting that if you desire to create a single point cloud from multiple views, this can be easily accomplished by appending the new point clouds to the previous ones. For example, if you have a 3×15000 point cloud and a 3×20000 point cloud from two different views of the same scene, they can be combined into a single 3×35000 point cloud.

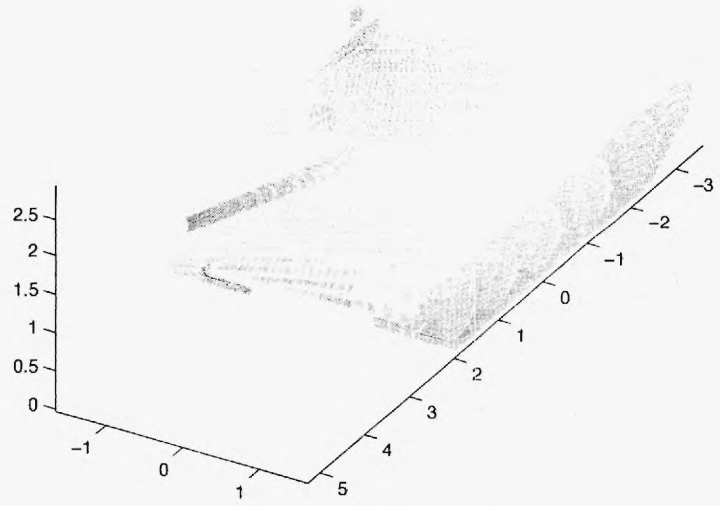
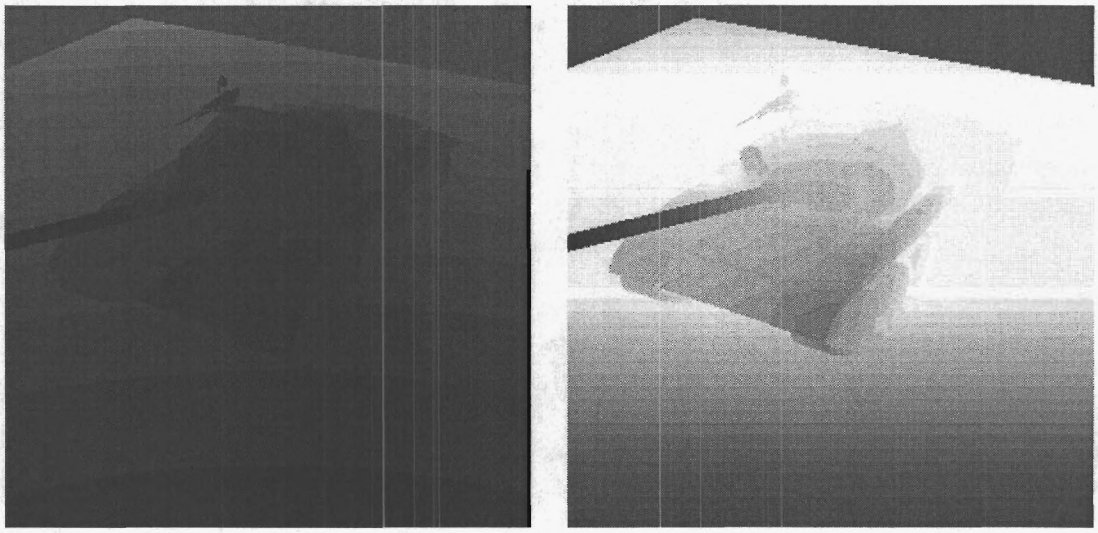


Figure 5.4: A typical point cloud.

5.3.3 Depth Buffer

If desired, the user can also create images that consist of the raw values pulled from the OpenGL depth buffering system after rendering the scene. This type of image is similar to the range image in that each pixel represents some notion of range, but the values will be arbitrary double precision floating point values. This tends to be useful for visualizing data sets since depth is treated linearly without converting to true range value. The effect of this is that the default colormap will not obscure certain image features if the minimum and maximum ranges are set too far apart. Rendering is also faster since there is no conversion to range values, allowing the GUIs to be more responsive to changes and simulations to run faster. It is beneficial to work in this mode when setting up a scene. Figure 5.5 shows a side-by-side comparison of a range image and a depth buffer image with larger then necessary minimum and maximum range difference.



(a) (b)

Figure 5.5: Comparison of a range image and depth buffer image when minimum and maximum ranges are set at 0.1 and 100, respectively.

Appendix B

J.A. Dixon and A.D. Lanterman, "Toward Practical Pattern-Theoretic ATR Algorithms for Infrared Imagery," *Automatic Target Recognition XVI*, SPIE Vol. 6234, Ed: F.A. Sadjadi, April 2006, pp. 212–220.

Toward practical pattern-theoretic ATR algorithms for infrared imagery

Jason H. Dixon and Aaron D. Lanterman

Center for Signal and Image Processing
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332, USA

ABSTRACT

Techniques for automatic target recognition (ATR) in forward-looking infrared (FLIR) data based on Grenander's pattern theory are revisited. The goal of this work is to unify two techniques: one for multi-target detection and recognition of pose and target type, and another for structured inference of forward-looking infrared (FLIR) thermal states of complex objects. The multi-target detection/recognition task is accomplished through a Metropolis-Hastings jump-diffusion process that iteratively samples a Bayesian posterior distribution representing the desired parameters of interest in the FLIR imagery. The inference of the targets' thermal states is accomplished through an expansion in terms of "eigentanks" derived from a principle component analysis over target surfaces. These two techniques help capture much of the variability inherent in FLIR data. Coupled with future work on rapid detection and penalization strategies to reduce false alarms, we strive for a unified technique for FLIR ATR following the pattern-theoretic philosophy that may be implemented for practical applications.

Keywords: automatic target recognition, ATR, infrared, FLIR, pattern theory

1. INTRODUCTION

The problem of detecting and classifying objects of interest in images has been extensively studied, producing many viable techniques. Many ATR systems that are in use today tend to divide the process of recognition into separate stages. These include target detection, feature extraction, clutter rejection, classification, and possibly other stages, depending on the nature of the algorithm. Infrared imagery is challenging because, in addition to geometric variability, we must also deal with the thermal variability related to target heat signatures changing under different operational and environmental conditions. This is analogous to the challenges posed by varying illumination in visual-band imagery.

In the mid-1990s, an effort was initiated at Washington University in St. Louis to develop pattern-theoretic algorithms for ATR in infrared imagery. The fruits of that work included a process for detecting and classifying multi-target scenes consisting of known target types, estimating the thermal signature characteristics of targets of interest, and ideas for the inference of targets of unknown type (or shape, depending on how you view the problem). Most of these techniques remained disjoint and were never fused into a unified ATR framework. In this work, we seek to unify two of these methods: multi-target detection/recognition and thermal state estimation.

1.1. Pattern theory

Ulf Grenander's work in pattern theory is the motivating force behind our framework for automatic target recognition.[1,2] While most computer vision and object recognition techniques focus on the separate stages of recognition (feature extraction, segmentation, classification, etc), the pattern-theoretic framework seeks to unify these separate concepts into a single process such that all steps are performed jointly. The detection/recognition process is performed directly on the data itself, in the hopes that the information loss that may arise from traditional preprocessing schemes may be avoided.

In following the pattern theory philosophy, we must first define the elements within a FLIR image. The patterns that we are interested in are built from templates. These templates may undergo transformations

Tel: 404-385-2548. Fax: 404-894-8363. E-mail: jdixon@ece.gatech.edu, lanterma@ece.gatech.edu

such as translations, rotations, changes of scale, or any other action that can be represented mathematically. To determine the transformations present in the imagery acquired from the FLIR sensor, we must be able to synthetically create similar imagery; thus the tasks of pattern synthesis and pattern recognition are linked in this framework. Continuing with this pattern-theoretic terminology, we will refer the objects of interest within the imagery as "generators". A "configuration" will denote the set of generators that make up the scene.

1.2. Representation of complex scenes

In this study, the set of generators in a scene configuration will consist of an unknown number of vehicles. This is the image seen by the FLIR sensor. Each generator will contain knowledge about the object it represents, which in this case includes position, orientation, type, and thermal profile. A single generator g representing a ground-based target in the set of generators $\mathcal{G}$ will be part of the configuration space $\mathcal{C}^1 = \Re^2 \times [0, 2\pi) \times \mathcal{A} \times \boldsymbol{\alpha}$ which defines a ground-based position and orientation, a generator class representing the type of target (e.g. $\mathcal{A}$={M2,M60,T62,...}), and a set of thermal parameters $\boldsymbol{\alpha}$. A scene with N targets lives in a space $\mathcal{C}^N$. Since the number of targets is not known in advance, the full parameter space is a union $\mathcal{C} = \bigcup_{N=0}^{\infty} \mathcal{C}_N$.

2. FLIR STATISTICS

Our analysis of FLIR data sets for ATR purposes is based on a Bayesian framework. We start with a likelihood function that models FLIR sensor statistics, and use it to compare scenes synthesized from hypothesized configurations with the collected data. The likelihood is combined with prior information to form a Bayesian posterior distribution. We consider uniform priors over position and orientation. In practice, we implicitly introduce the prior information that two targets may not occupy the same space by not considering such scenes.

2.1. Gaussian likelihood model

Our earlier FLIR ATR studies assumed a likelihood function based on Poisson statistics.[3] The FLIR sensor was taken to be a CCD detector producing Poisson distributed data with means proportional to the radiant intensities of the objects in the sensor's field of view. Using such a model assumes the sensor is calibrated to give specific photon counts. While this is often true in astronomical imaging, it is usually not the case for operational FLIR sensors. Hence, we switch to a Gaussian model similar to the one discussed by Koskal et al.[4] This model assumes the measured temperature is related to the true temperature of the objects present in the scene by Gaussian distributed noise consisting of a combination of thermal noise and shot noise. We treat the temperature measurements at each pixel to be independent of all other pixels, so the loglikelihood function becomes

$$L_{IR}(\mathbf{d}|\boldsymbol{\mu}) = \sum_n \left\{ -\log \sqrt{2\pi(NE\Delta T)^2} - \frac{(d(n) - \mu(n))^2}{2(NE\Delta T)^2} \right\}, \tag{1}$$

where $\boldsymbol{\mu}$ is an ideal noiseless image, $\mathbf{d}$ is the measured data, and $NE\Delta T$ is the FLIR's noise-equivalent temperature difference. The summation is computed over all pixels n in a given data image. In this study, we are only concerned with how the loglikelihood changes with different scene configurations, so we can ignore terms that do not depend on $\boldsymbol{\mu}$ and simply reduce the loglikelihood function per pixel to the squared error between the measured data and a hypothetical uncorrupted image $\boldsymbol{\mu}$.

2.2. Gibbs posterior distribution

Given an estimated configuration state c, the likelihood and the prior combine to form a Gibbs posterior distribution of the form

$$\pi(c|\mathbf{d}) \propto \exp[H(c|\mathbf{d})], \tag{2}$$

where $H(c|\mathbf{d}) = L(\mathbf{d}|c) + P(c)$ is the logposterior created by summing the loglikelihood $L(\mathbf{d}|c)$ and the logprior $P(c)$. $L(\mathbf{d}|c) = L_{IR}[\mathbf{d}|\text{render}(c)]$ where render(c) is the process of obtaining an uncorrupted image $\boldsymbol{\mu}$ from a scene configuration c through perspective projection and obscuration. This distribution will represent how closely related the hypothesized configuration is to the data image. This distribution is sampled by a type of reversible jump Markov chain Monte Carlo routine called a jump-diffusion process, described in Section 4.

3. REPRESENTING VARIABILITY IN INFRARED IMAGERY

3.1. Eigentanks

We approach the modeling of the thermal variations of targets from the mindset of empirical statistics and construct prior distributions on the radiant intensities of target facets.[5–8] By simulating a large number of radiance measurements, taken while varying environmental and internal heating parameters over reasonable ranges, we generate a population of radiance profiles to which we apply principal component analysis. For simulating radiances, we employ the PRISM software originally developed by the Keweenaw Research Center at Michigan Technological University.* Assuming a Gaussian model, the first few eigenvectors - here called "eigentanks" - provide a parsimonious representation of the covariance.†

Suppose the surface of the CAD model of the tank is divided into I regions, with the intensity assumed constant across each region, and that we are employing J basis functions. Let A_i denote the surface area of region i and λ_i represent the intensity of region i. We employ representations of the form $\lambda_i = \sum_j \alpha_j \Phi_{ij} + m_i$, where m_i is the mean of region i, Φ_{ij} is eigentank j at region i, and γ_j is the eigenvalue associated with eigentank j. The α_j's are expansion coefficients.

To generate the eigentank models, we first synthesize a large database of N radiance maps, written as a vectors $\boldsymbol{\lambda}^{DB}(n) = [\lambda_1^{DB}(n), \ldots, \lambda_I^{DB}(n)]^T \in \Re_+^{I\times 1}$ for $n = 1, \ldots, N$. The radiance maps are simulated under a wide range of conditions, both meteorological (solar irradiance, wind speed, relative humidity, etc.) and operational (vehicle speed, engine speed, gun fired or not, etc.) to yield a wide variety of sample thermal states.

We compute the empirical mean $\mathbf{m} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\lambda}^{DB}(n)$ and covariance

$$\mathbf{K} = \frac{1}{N}\sum_{n=1}^{L} [\boldsymbol{\lambda}^{DB}(n) - \mathbf{m}][\boldsymbol{\lambda}^{DB}(n) - \mathbf{m}]^T. \tag{3}$$

We seek the eigenvalues γ_i and the eigentanks Φ_{ij} that satisfy

$$\gamma_j \Phi_{ij} = \sum_l K_{il}\Phi_{lj}A_l. \tag{4}$$

Notice the weighting by the surface measure. Writing the A's as a diagonal matrix $\mathbf{A}$ and the eigentanks as $\boldsymbol{\Phi}_j = [\Phi_{1j}, \ldots, \Phi_{Ij}]^T \in \Re^{J\times 1}$, we can express (4) as $\gamma_j \boldsymbol{\Phi}_j = \mathbf{KA}\boldsymbol{\Phi}_j$. The eigentanks $\boldsymbol{\Phi}_j$ and eigenvalues γ_j can be readily found via standard numerical routines.

3.2. Logposterior for rigid targets

This discussion will follow the derivation found in Lanterman et al.[8] but will employ our Gaussian data likelihood instead of a Poisson likelihood. Consider a collected data set $\mathbf{d}$. Let N_i denote the number of pixels in region i, as seen by the detector, and $D_i = \sum_{k\in R_i} d(k)$ be the sum of data pixels in region i. Conditioned on the α_j's, $d(k) \sim Gaussian(\lambda_i, (NE\Delta T)^2)$ for $k \in R_i$. In accordance with the principal component analysis discussed in the preceding section, a Gaussian prior is placed on the α's, with variances given by the eigenvalues found from the analysis. Dropping terms independent of α, the logposterior for the pixels on target in terms of the expansion coefficients is

$$H(\alpha|D) \approx -\sum_i \sum_{k\in R_i} \frac{(\lambda_i - d(k))^2}{2(NE\Delta T)^2} - \sum_j \frac{\alpha_j^2}{2\gamma_j} \tag{5}$$

$$= -\sum_i \sum_{k\in R_i} \frac{\lambda_i^2 - 2\lambda_i d(k) + d^2(k)}{2(NE\Delta T)^2} - \sum_j \frac{\alpha_j^2}{2\gamma_j} \tag{6}$$

$$= -\sum_i \frac{N_i\lambda_i^2 - 2\lambda_i \sum_{k\in R_i} d(k) + \sum_{k\in R_i} d^2(k)}{2(NE\Delta T)^2} - \sum_j \frac{\alpha_j^2}{2\gamma_j}. \tag{7}$$

*PRISM was sold and maintained by ThermoAnalytics, Inc., P.O. Box 66, Calumet, MI 49913, web: www.thermoanalytics.com. It has been replaced by the MuSES infrared signature prediction software

†The PRISM databases and resulting principle component models employed in the experiments discussed here were created by Dr. Matthew Cooper.[5–7]

Incorporating $\lambda_i = \sum_j \alpha_j \Phi_{ij} + m_i$ and taking the derivative with respect to each α_j, we obtain equations of the form

$$\frac{\partial}{\partial \alpha_j} H(\alpha|D) = -\sum_i \frac{2N_i\lambda_i \frac{\partial}{\partial \alpha_j}\lambda_i - 2\frac{\partial}{\partial \alpha_j}\lambda_i \sum_{k\in R_i} d(k)}{2(NE\Delta T)^2} - \frac{\alpha_j}{\gamma_j} \quad (8)$$

$$= -\sum_i \frac{N_i(\sum_k \alpha_k \Phi_{ik} + m_i)\Phi_{ij} - \Phi_{ij} D_i}{(NE\Delta T)^2} - \frac{\alpha_j}{\gamma_j} \quad (9)$$

To maximize the logposterior, we must satisfy these J necessary conditions:

$$-\sum_i \frac{N_i(\sum_k \alpha_k \Phi_{ik} + m_i)\Phi_{ij} - \Phi_{ij} D_i}{(NE\Delta T)^2} - \frac{\alpha_j}{\gamma_j} = 0, \forall j \quad (10)$$

$$-\sum_k \alpha_k \sum_i \frac{N_i \Phi_{ik}\Phi_{ij}}{(NE\Delta T)^2} - \sum_i \frac{N_i \Phi_{ij} m_i - \Phi_{ij} D_i}{(NE\Delta T)^2} - \frac{\alpha_j}{\gamma_j} = 0, \forall j \quad (11)$$

$$-\sum_k \alpha_k \sum_i [N_i \Phi_{ik}\Phi_{ij}] - \sum_i \Phi_{ij}[N_i m_i - D_i] - \frac{(NE\Delta T)^2}{\gamma_j}\alpha_j = 0, \forall j. \quad (12)$$

Fortunately these are linear equations, conveniently expressed in matrix form:

$$\left(\begin{bmatrix} \sum_i N_i \Phi_{i1}^2 & \cdots & \sum_i N_i \Phi_{iJ}\Phi_{i1} \\ \vdots & & \vdots \\ \sum_i N_i \Phi_{i1}\Phi_{iJ} & \cdots & \sum_i N_i \Phi_{JJ}^2 \end{bmatrix} + diag \begin{pmatrix} \frac{(NE\Delta T)^2}{\gamma_1} \\ \vdots \\ \frac{(NE\Delta T)^2}{\gamma_J} \end{pmatrix} \right) \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_J \end{bmatrix} = \begin{bmatrix} \sum_i \Phi_{i1}(D_i - N_i m_i) \\ \vdots \\ \sum_i \Phi_{iJ}(D_i - N_i m_i) \end{bmatrix}. \quad (13)$$

For a given target pose, these equations allow us to compute approximate MAP estimates for the α_j's in closed form, which we denote as $\hat{\alpha}_j$. Note that $\hat{\alpha}_j$ changes with different poses, as the MAP estimate adjusts to best match the data under the constraint of the eigentank model. For a multiple target scene, we perform a similar calculation for each target.

4. INFERENCE BY METROPOLIS-HASTINGS JUMP-DIFFUSIONS

The full posterior distribution represented by (2) quantifies the relationship between the data and all possible hypothesized scenes, and is thus a function of several pose, type, and thermal state parameters with a dimension that changes with the number of targets in the hypothesized scene. As shown in the previous section, we can easily estimate the thermal state parameters given a particular pose. To estimate the parameters of interest, we need a way to sample from this posterior distribution that can also adjust the number of targets, since we do not know this information in advance. We follow the jump-diffusion framework presented by Lanterman et al.,[3] Miller et al.,[9] and Srivastava et al.[10] The algorithm is designed around a reversible jump Markov process that accounts for the continuous and discrete aspects of the ATR problem. The discrete components are handled by "jumping" from one inference task to another, deciding whether to add a target, remove a target, or changing a target's type. These choices will henceforth be called "birth", "death", and "metamorph", respectively. The continuous aspects, namely the inference of the pose parameters, are refined via a diffusion process.

The decision process used when determining whether or not to accept a jump involves random sampling and Metropolis-Hastings acceptance/rejection. For birth and metamorph jumps, a representative set of candidate locations and orientation angles are chosen, respectively. For death jumps, candidates are chosen by removing a single target from the hypothesized configuration. The posterior probabilities are computed at each of these candidates and one candidate is randomly chosen with a probability proportional to its posterior probability. In practice, one candidate posterior probability typically "swamps" the others so it often *appears* as though the candidate with maximum probability is automatically chosen. The Metropolis-Hastings algorithm accepts the chosen candidate with probability

$$\beta(c_{orig}, c_{prop}) = \min\left(\frac{\pi(c_{prop})r(c_{prop}, c_{orig})}{\pi(c_{orig})r(c_{orig}, c_{prop})}, 1\right), \quad (14)$$

where c_{prop} is the proposed state of the configuration while c_{orig} is the current state. The functions $r(c_{prop}, c_{orig})$ and $r(c_{orig}, c_{prop})$ are the transition probabilities; see Lanterman et al.[3] for details. The function $\pi(c)$ is the probability of being in state c, which in this case is derived from the logposterior $H(c|\mathbf{d})$ for that state.

The choice of which type of jump to perform is determined probabilistically by a prior distribution based on the number of hypothesized targets in the configuration. As typical with continuous-time Markov processes, the time between jumps is exponentially distributed. During these intervals between jumps, the diffusion process takes over and perturbs the continuous pose parameters by small amounts to better align the hypothesized targets with the data. Diffusions are accomplished using the Langevin stochastic differential equation:

$$d\mathbf{C}_N(\tau) = \nabla_{\mathbf{C}_N} H[\mathbf{C}_N(\tau)|\mathbf{d}]d\tau + \sqrt{2}dW_N \tag{15}$$

where W_N is a Wiener process and $H[\mathbf{C}_N(\tau)|\mathbf{d}]$ is the logposterior associated with the configuration parameter vector $\mathbf{C}_N$, which contains the configuration parameters for N targets of fixed classes. The time index τ refers to a unit of time within the diffusion interval. Once (15) is discretized, it can simply be thought of as a discrete time index such that a finite number of diffusions will occur between jumps, and that number is an exponential random variable.

For a more detailed analysis of theory behind jump-diffusions in general, please refer to the aforementioned works by Lanterman et al.,[3] Miller et al.,[9] and Srivastava et al.[10]

5. IMPLEMENTATION

This analysis was performed on an Apple Macintosh G4 computer running MATLAB 7. Objects were rendered from faceted tank models using the OpenGL three-dimensional graphics application programming interface (API). OpenGL performs the transformation $(\bar{x}, \bar{y}, \bar{z}) \rightarrow (\bar{x}/\bar{z}, \bar{y}/\bar{z})$, taking a three-dimensional hypothesized scene and turning it into a two-dimensional image through perspective projection and obscuration. Our example configurations consist of a combination of M60 and T62 faceted tank models placed over an infrared background image. The background image was included to give a sense of what a true infrared scene would look like, but no effort was made to relate the viewing parameters of the background to the viewing parameters of the rendered images. The rendered image was corrupted by Gaussian noise and bad pixels. Targets are assumed to be resting on a ground plane with unknown position $(x, y)^{\ddagger}$, unknown orientation angle θ, unknown type $a \in \mathcal{A} =$ (M60, T62), and unknown thermal state represented by the expansion coefficients for each eigentank $[\alpha_i, \cdots, \alpha_J]^T$.

To determine the N_i and D_i terms required when computing the expansion coefficients, we used a "paint by numbers" technique.[8] Each target in a configuration is rendered with a set of increasing, yet disjoint, region numbers so that each intensity region is colored by a different number. One can easily compute the N_i by counting the pixels of a common region number and D_i by summing the corresponding data pixels. After computing the expansion coefficients, the inferred intensities can be used to color in the image of region numbers. We consider the background to be of constant intensity equal to the average background intensity of the data. This final image is considered the hypothesized true scene and is compared to the data set through the loglikelihood function.

The Gibbs-form posterior distribution is explored by sampling the space of possible configurations with respect to the parameters of interest, which involves rendering hypothesized scenes during any birth, death, or metamorph move. In determining the acceptance probability for any move, we obtained good results by assuming that the forward and reverse transition probabilities are equal and simply comparing the proposal and original posterior probabilities of the hypothesized configuration scenes. For a more intuitive sense of the implementation of the jump-diffusion process, see Figure 1.

The derivative needed in solving the Langevin SDE (15) was computed with a finite difference approximation:

$$\frac{\partial H(c|\mathbf{d})}{\partial c_p} \approx \frac{H(\ldots, c_p + \delta, \ldots|\mathbf{d}) - H(\ldots, c_p - \delta, \ldots|\mathbf{d})}{2\delta} \tag{16}$$

where c_p is an element of the configuration c, δ is some small deviation of the parameter c_p, and the ellipses indicate the remaining parameters are held fixed.

[‡]We use tildes earlier to indicate that the coordinate system is transformed according to the camera position and orientation.

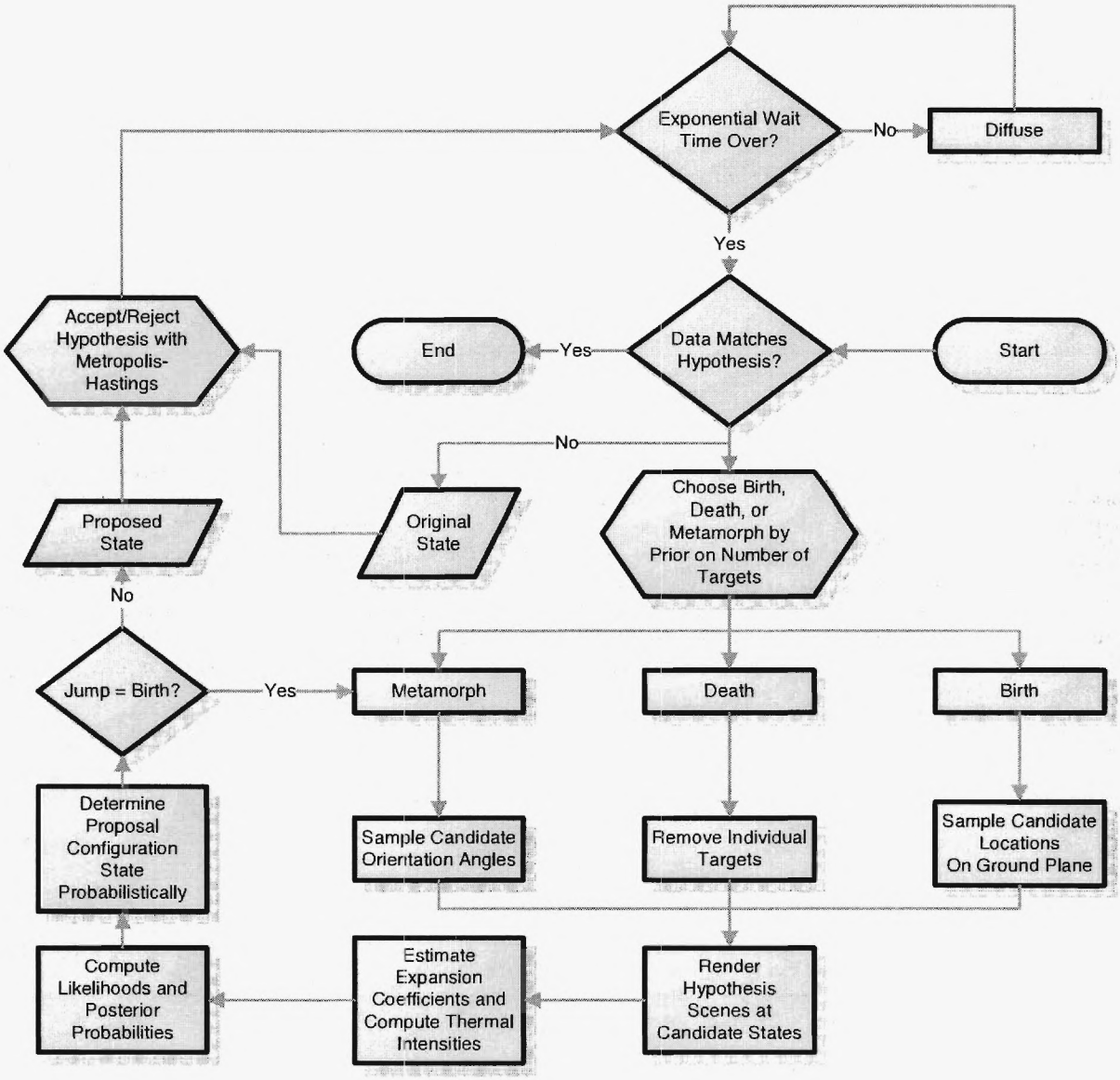


Figure 1. Simple block diagram for implementing the jump-diffusion process for ATR in infrared data.

6. RESULTS

6.1. Jump-diffusion experiments

Results from a jump-diffusion simulation are shown in Figure 2. Figure 2(a) shows the initial FLIR dataset used in this simulation which consists of four tanks. The top-most and bottom-most tanks are both M60s and the remaining two are T62s. Each was initialized with a different position, orientation, and thermal state. We assume the camera viewing parameters are known. The algorithm begins by searching over the configuration space for the best set of parameters for a single new target. In the subsequent images, the white tanks represent the estimated configuration at that point in the simulation. Figure 2(b) shows that the first tank located is the M60 that is positioned closest to the FLIR sensor. Since this tank has the greatest number of pixels on target, it makes sense that the algorithm would choose that position for a tank to achieve the greatest gain in likelihood.

The next few images shown in Figures 2(c)-2(e) show how the algorithm subsequently detects and places new targets over the existing targets in the data set. Between the birth, death, and metamorph moves, the algorithm diffuses over the existing targets in an attempt to refine their pose parameters. The estimated thermal emission profiles of the hypothesized targets change as the diffusions take place due to the adjustments of pose changing the overlap with the corresponding target in the data image.

Figures 2(f) and 2(g) are interesting because they demonstrate the flexibility of the jump-diffusion process. A metamorph move occurs before the rightmost T62 fully diffuses over the T62 in the data set, and the best orientation angle turns out to be in the opposite direction (the estimated tank no longer points away from the FLIR sensor as shown in the data). Once the diffusion allows the estimated T62 to noticeably cover the T62 in the data image, another metamorph move brings it into the appropriate alignment, as shown in Figure 2(g). The final estimate is shown in Figure 2(h). All of the targets are aligned with the corresponding targets in the data image. The estimated thermal emissive profiles also match those found in the data.

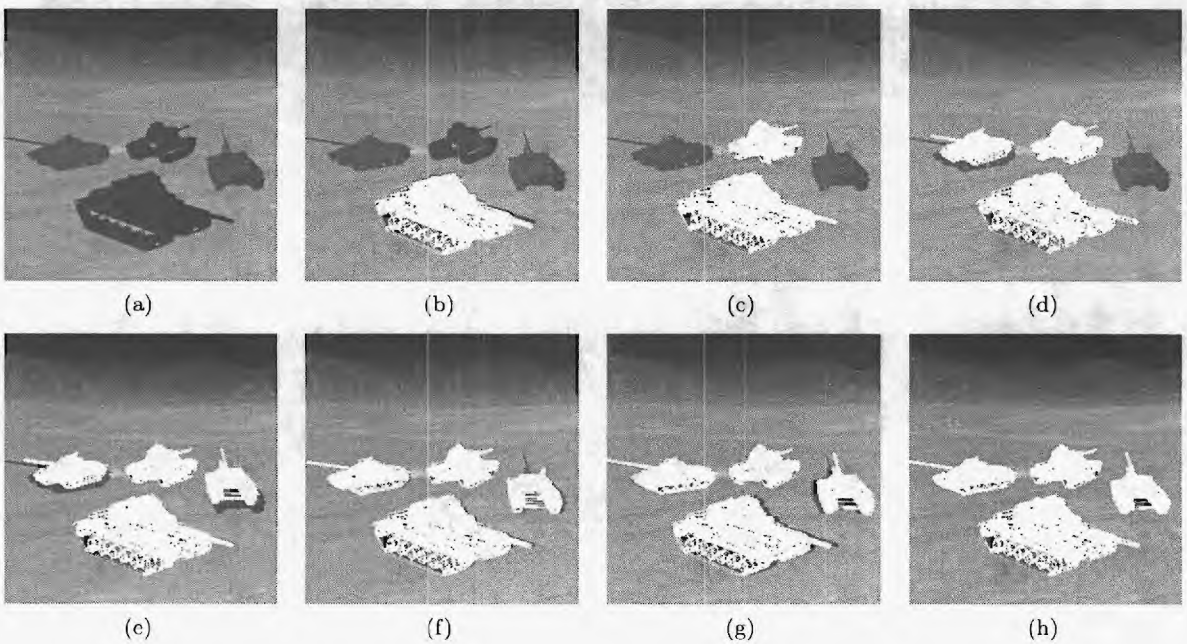


Figure 2. Screen captures from iterations of a jump-diffusion process for FLIR ATR.

6.2. Phantom targets

With the incorporation of the inference of target thermal signatures in a given configuration, our jump-diffusion algorithm suffers from a potential increase in false alarms when initially detecting targets. This can be attributed

to the amount variability that our current eigentank model is capable of capturing, which includes thermal states that are generally not attainable by actual targets. If we are rendering a target model over a segment of the data image that contains a target of the same type, and if pose parameters match up correctly, then the inferred thermal states will also match the true values since intensity regions will overlap properly. If any of these conditions are not met, then the computation of the expansion coefficients will either include matching the hypothesized target's intensity regions with the wrong regions of targets in the data or matching intensity regions with background. If the background includes clutter that contains thermal characteristics similar to those of known targets, then algorithm may choose a thermal profile that blends into the background. If this happens during a birth move in the jump-diffusion process, a "phantom target" can appear in the hypothesized scene. Since the likelihoods tend to increase with the existence of these phantom targets, it becomes more difficult for the jump-diffusion algorithm to remove them later on through a death movement.

An example of this can be seen in Figure 3. Here, we simply computed expansion coefficients at four different positions over an infrared image that contained no targets. The purpose was to see which facets of the resulting targets had thermal characteristics similar to the background data. As seen in the image, some target features appear as they would on a typical data set while other features blend into the background to the point where the tank almost disappears.

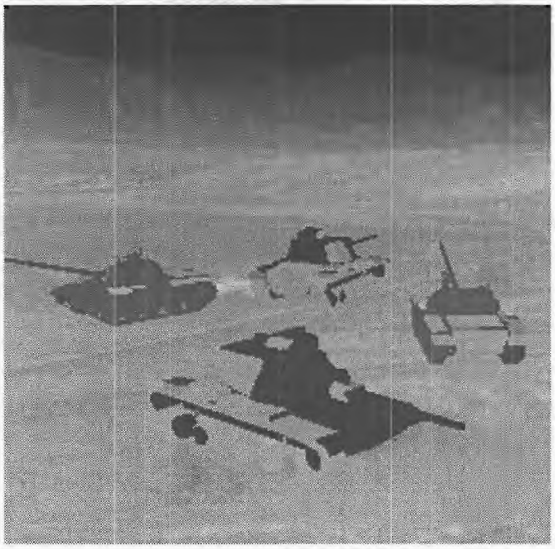


Figure 3. Tanks with radiance profiles derived from background emissions.

7. CONCLUSIONS AND FUTURE AREAS OF RESEARCH

We have discussed the unification of two pattern-theoretic concepts in the realm of ATR for infrared data. Combining the jump-diffusion ATR algorithm with thermal state estimation, we can perform ATR tasks with data sets having a great deal of variability. One can visualize the usefulness of estimating the thermal characteristics of a target of interest. Knowing these values and the corresponding radiance regions can allow ATR systems to predict the state of the target of interest (i.e. if it is moving, if it has just fired, etc.). There are, however, a number of challenges to resolve before this ATR procedure can be viewed as practical. These include how the algorithm detects new targets, the need for stopping conditions so the algorithm knows when to terminate, a better method of determining the Langevin SDE (15) numerically, and a likelihood penalization strategy to avoid detecting a target over background information.

Even though targets can be found over time via our current birth strategy, if the viewing grid is sampled finely enough, this process does not make intelligent use of the available information. Guessing locations to

search and then rendering fully detailed CAD models at those locations is computationally expensive and does not guarantee that all targets will be detected. The algorithm must also deal with partial false alarms, which occur when a new target only partially overlaps with one that exists in the data set. In theory, the diffusion process should account for these situations, but in practice it can sometimes have difficultly when refining the pose parameters. A few potential rapid detection schemes were investigated by Lanterman et al.[11]

In practice, we must add a criterion to determine when the algorithm has "converged." Assuming that the data can be represented by a hidden set of configuration parameters, there is nothing to tell the algorithm that the current set of estimated configuration parameters are close to these hidden ones.

Two more issues arise when computing the Langevin SDE during a diffusion: the choice of stepsizes for the derivative computation and the choice of stepsizes for the diffusions themselves. In the previously discussed experiments, both of these were determined empirically through a trial-and-error approach that yielded the best adjustments to the configuration parameters. In practice, these need to be determined in an automated fashion because they depend on the types of targets in the scene and the scene's viewing parameters. We consider alternative algorithms that facilitate local parameter search via Metropolis-Hastings or Gibbs style sampling in a small region around the current parameter estimate. Using these schemes, we can adjust the configuration parameters so that they adjust the targets by individual pixel values, leading to faster convergence rates.

With the additional variability that we can now model with the eigentank expansion incorporated into the jump-diffusion process, there is a need to restrict that variability to actual target types. As shown in Section 6.2, matching a target against a portion of background with similar thermal signatures becomes even more likely. In the future, we plan to examine different types of penalties[12] that can be used with the likelihood function to reduce these false alarms.

Finally, it should be noted that these algorithms have not been tested with real infrared data. Before we can do so, we need to solve the problem of calibration between our models and real infrared images.

ACKNOWLEDGMENTS

This work was sponsored by the Air Force Office of Scientific Research (AFOSR) grant F49620-03-1-0340 .

REFERENCES

1. U. Grenander and M. I. Miller, "Representations of knowledge in complex systems," *Journal of the Royal Statistical Society. Series B (Methodological)* **56**(4), pp. 549–603, 1994.
2. U. Grenander, *Elements of Pattern Theory*, Johns Hopkins University Press, Baltimore, MD, 1996.
3. A. D. Lanterman, M. I. Miller, and D. L. Snyder, "General Metropolis-Hastings jump diffusions for automatic target recognition in infrared scenes," *Optical Engineering* **36**(4), pp. 1123–37, 1997.
4. A. E. Koksal, J. H. Shapiro, and M. I. Miller, "Performance analysis for ground-based target orientation estimation: FLIR/LADAR sensor fusion," *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers* **vol.2**, pp. 1240–4, (Pacific Grove, CA), 1999.
5. M. L. Cooper, A. D. Lanterman, S. Joshi, and M. I. Miller, "Representing the variation of thermodynamic state via principle component analysis," in *Proc. of the Third Workshop on Conventional Weapon ATR*, pp. 479–490, U.S. Army Missile Command, 1996.
6. M. L. Cooper, U. Grenander, M. I. Miller, and A. Srivastava, "Accommodating geometric and thermodynamic variability for forward-looking infrared sensors," in *Algorithms for Snythetic Aperture Radar IV*, E. Zelnio, ed., **SPIE Proc. 3070**, pp. 162–172, (Orlando, FL), 1997.
7. M. L. Cooper and M. I. Miller, "Information measures for object recognition," in *Algorithms for Synthetic Aperture Radar Imagery V*, E. Zelnio, ed., **SPIE Proc. 3370**, pp. 637–645, SPIE, (Orlando, FL), 1998.
8. A. D. Lanterman, "Bayesian inference of thermodynamic state incorporating Schwarz-Rissanen complexity for infrared target recognition," *Optical Engineering* **39**(5), pp. 1282–1292, 2000.
9. M. I. Miller, U. Grenander, J. A. O'Sullivan, and D. L. Snyder, "Automatic target recognition organized via jump-diffusion algorithms," *IEEE Transactions on Image Processing* **6**(1), pp. 157–74, 1997.

10. A. Srivastava, U. Grenander, G. R. Jensen, and M. I. Miller, "Jump-diffusion Markov processes on orthogonal groups for object pose estimation," *Journal of Statistical Planning and Inference* **103**(1-2), pp. 15–37, 2002.
11. A. D. Lanterman, M. I. Miller, and D. L. Snyder, "Representations of shape for structural inference in infrared scenes," in *Automatic Object Recognition VII*, F. A. Sadjadi, ed., **Proc. SPIE 3069**, pp. 257–268, (Orlando, FL), 1997.
12. A. D. Lanterman, "Schwarz, Wallace, and Rissanen: Intertwining themes in theories of model order estimation," *International Statistical Review* **Vol. 69**(No. 2), pp. 185–212, 2001.

Appendix C

J.A. Dixon and A.D. Lanterman, "Information-Theoretic Bounds on Target Recognition erformance from Laser Radar Data," *Automatic Target Recognition XVI*, SPIE Vol. 6234, Ed: F.A. Sadjadi, April 2006, pp. 394–403.

Information-theoretic bounds on target recognition performance from laser radar data

Jason H. Dixon and Aaron D. Lanterman

Center for Signal and Image Processing
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332, USA

ABSTRACT

Laser radar systems historically offer rich data sets for automatic target recognition (ATR). ATR algorithm development for laser radar has focused on achieving real-time performance with current hardware. Our work addresses the issue of understanding how much information can be obtain from the data, independent of any particular algorithm. We present Cramer-Rao lower bounds on target pose estimation based on a statistical model for laser radar data. Specifically, we employ a model based on the underlying physics of a coherent-detection laser radar. Most ATR algorithms for laser radar data are designed to be invariant with respect to position and orientation. Our information-theoretic perspective illustrates that even algorithms that do not explicitly involve the estimation of such nuisance parameters are still affected by them.

Keywords: automatic target recognition, ATR, Cramer-Rao bounds, laser radar, information theory

1. INTRODUCTION

As the number of target recognition algorithms increases, so does the need for accurate ways to compare the performance of one algorithm with another. Without a measure for performance, users of target recognition systems will have no way of identifying the superiority of one algorithm compared to another, or determining if the sensor in use needs to be improved to achieve better target recognition results. It is also useful to know if there is a fundamental limit on the ability to estimate a target's parameter of interest from data from a sensor. In this study, we consider targets imaged through laser radars. By reformulating the target recognition problem as a deterministic parameter estimation problem, we can apply the Cramer-Rao lower bound (CRLB) to determine this measure of performance and these fundamental limits.

A number of studies have used information-theoretic bounds for the purposes of performance estimation or the accuracy in estimating parameters of interest. One such study, performed by Koskal et al.,[1] used performance bounds to determine pose estimation accuracy from forward-looking infrared (FLIR) and laser radar (LADAR). In these experiments, Cramer-Rao bounds were compared to Hilbert-Schmidt bounds on the mean squared error of estimating orientation as a function of noise.[2] In a study performed by Jain et al., a number of information-theoretic tools were used to determine bounds on target detection performance in images with noise and clutter.[3] The bounds noted in that study included Kullback-Leibler and Chernoff distances. The study considered the cases without nuisance parameters and with the nuisance parameter of orientation. Our work focuses on standard deviation bounds on estimating pose parameters, although most ATR algorithms consider them to be nuisance parameters. We are considering the pose parameters directly because of their importance in aim point selection or other scenarios where it is important to identify specific points on a target. We investigate how the bounds change as we vary the distance, position, and orientation of the targets. A similar study using CRLBs was performed by Gerwe et al.[4,5] to determine bounds on orientation when viewing satellites from ground-based optical sensors.

All imagery required in our CRLB study is synthetic by nature, so we are not bound to using one specific collection of laser radar data sets. We can generate scenes with any viewing parameters and any combination of targets. This allows us to see how the CRLBs change as the scene parameters change.

Tel: 404-385-2548. Fax: 404-894-8363. E-mail: jdixon@ece.gatech.edu, lanterma@ece.gatech.edu

This paper begins with a discussion of our LADAR statistical model in Section 1.1. We then consider the CRLB and its derivation for our LADAR signal model in Section 1.2. Next, we proceed into the implementation and the experimental discussion in Sections 1.3 and 2 respectively. Then, we discuss some of the results and what information we gain from performing this CRLB analysis in Sections 3 - 3.4. Finally, we will conclude and offer some suggestions for future work in this area in Section 4.

1.1. LADAR statistical model

For LADAR images, we employ a coherent-detection likelihood function that incorporates single-pixel noise statistics developed by Shapiro and Green[6,7] and used in a multi-target ATR study for LADAR data.[8] If $\mathbf{d}$ is an image of measured range values and $\boldsymbol{\mu}$ is an image of uncorrupted "true" range values, then the loglikelihood function is given by

$$L_{LR}(\mathbf{d};\boldsymbol{\mu}) = \sum_n \log\left\{[1 - Pr_A(n)]\frac{1}{\sqrt{2\pi\sigma^2(n)}}\exp\frac{-\{d(n)-\mu(n)\}^2}{2\sigma^2(n)} + \frac{Pr_A(n)}{R_{amb}}\right\}, \tag{1}$$

where $\sigma(n)$ is the local range accuracy for pixel n given by

$$\sigma(n) = \frac{R_{res}}{\sqrt{\mathrm{CNR}(n)}}, \tag{2}$$

and $Pr_A(n)$ is the probability of anomalous measurement for pixel y given by

$$Pr_A(n) = \frac{\log(N) - \frac{1}{N} + 0.557}{\mathrm{CNR}(n)}. \tag{3}$$

R_{amb} is the range ambiguity interval and N is the number of range resolution bins given by

$$N = \frac{R_{amb}}{R_{res}}. \tag{4}$$

R_{res} is the range resolution, and CNR is the carrier-to-noise ratio taken to be

$$\mathrm{CNR}(n) = \frac{\eta P_T \rho A_R}{h\nu B\pi}\epsilon_{opt}\epsilon_{het}\frac{e^{-2\alpha\mu(n)}}{\mu^2(n)}, \tag{5}$$

where α is the atmospheric extinction coefficient, ϵ_{opt} is the receiver's optical efficiency, ϵ_{het} is the receiver's heterodyne efficiency, η is the detector's quantum efficiency, $h\nu$ is the photon energy, ρ is the target reflectivity, B is the IF filter bandwidth, P_T is the peak transmitting power, and A_R is the receiver's aperture area. We will only consider the case with no anomalous measurements (i.e., $Pr_A = 0$) so the last term drops out of the loglikelihood function. It can now be reduced to

$$L_{LR}(\mathbf{d};\boldsymbol{\mu}) = \sum_n -\frac{1}{2}\log\left\{2\pi\sigma^2(n)\right\} - \frac{[d(n)-\mu(n)]^2}{2\sigma^2(n)}. \tag{6}$$

1.2. Derivation of the Cramer-Rao bound

The matrix Cramer-Rao bound for a vector of unbiased estimators is defined as the inverse of the Fisher information matrix (FIM) for that estimator vector, where each element of the FIM is defined as

$$F_{ij}(\boldsymbol{\Theta}) = E\left[\left(\frac{\partial}{\partial\Theta_i}\log p(D;\boldsymbol{\Theta})\right)\left(\frac{\partial}{\partial\Theta_j}\log p(D;\boldsymbol{\Theta})\right)\right], \tag{7}$$

where $p(d;\boldsymbol{\Theta})$ is the data loglikelihood, which is a function of the target parameters $\boldsymbol{\Theta} = [x, y, \theta]^T$. We use D in (7) to indicate that we would "plug in" a random variable for d. The coordinates x and y denote the target location on the ground plane, which is assumed to be flat, and θ is the orientation angle with respect to the axis

that points out of the ground plane. We treat the loglikelihood as a complicated, nonlinear function of $\boldsymbol{\Theta}$ that can be observed through the uncorrupted range image μ. It is best to think of $\mu(n)$ as $\mu(n;\boldsymbol{\Theta}) = \text{render}(\boldsymbol{\Theta})$ at pixel n in the resulting two-dimensional image created through obscuration and perspective projection of the three-dimensional scene consisting of a target with parameter vector $\boldsymbol{\Theta}$. To derive the Cramer-Rao lower bound, we begin by determining the derivative of the loglikelihood function with respect to the parameter vector $\boldsymbol{\Theta}$. To make the calculation simpler, we will compute $\sigma^2(n)$ for a given $\boldsymbol{\Theta}$, and approximate it as being *constant* with respect to small changes in $\boldsymbol{\Theta}$. With this approximation, the first term in (6) may be dropped, and the derivative can now be computed as follows:

$$\frac{\partial}{\partial\Theta_i} L_{LR}(\mathbf{D};\boldsymbol{\mu}) = \sum_n \frac{-2[D(n)-\mu(n;\boldsymbol{\Theta})]\frac{\partial}{\partial\Theta_i}\mu(n;\boldsymbol{\Theta})}{2\sigma^2(n)} \tag{8}$$

$$= \sum_n \frac{-[D(n)-\mu(n;\boldsymbol{\Theta})]\frac{\partial}{\partial\Theta_i}\mu(n;\boldsymbol{\Theta})}{\sigma^2(n)}. \tag{9}$$

Our sensor model treats the data scene $D(n)$ as a Gaussian random variable with a mean given by the uncorrupted range $\mu(n;\boldsymbol{\Theta})$ and variance $\sigma^2(n)$ for pixel n. Therefore, $E[D(n)] = \mu(n;\boldsymbol{\Theta})$. Using this relationship, the FIM becomes

$$F_{ij}(\boldsymbol{\Theta}) = E\left[\sum_n\left(\frac{-[D(n)-\mu(n;\boldsymbol{\Theta})]\frac{\partial}{\partial\Theta_i}\mu(n;\boldsymbol{\Theta})}{\sigma^2(n)}\right)\sum_{\tilde{n}}\left(\frac{-[D(\tilde{n})-\mu(\tilde{n};\boldsymbol{\Theta})]\frac{\partial}{\partial\Theta_j}\mu(\tilde{n};\boldsymbol{\Theta})}{\sigma^2(\tilde{n})}\right)\right] \tag{10}$$

$$= E\left[\sum_n\sum_{\tilde{n}}\left(\frac{D(n)-\mu(n;\boldsymbol{\Theta})}{\sigma^2(n)}\right)\left(\frac{D(\tilde{n})-\mu(\tilde{n};\boldsymbol{\Theta})}{\sigma^2(\tilde{n})}\right)\frac{\partial}{\partial\Theta_i}\mu(n;\boldsymbol{\Theta})\frac{\partial}{\partial\Theta_j}\mu(\tilde{n};\boldsymbol{\Theta})\right] \tag{11}$$

$$= \sum_n\sum_{\tilde{n}}\frac{E\left[\left(D(n)-\mu(n;\boldsymbol{\Theta})\right)\left(D(\tilde{n})-\mu(\tilde{n};\boldsymbol{\Theta})\right)\right]}{\sigma^2(n)\sigma^2(\tilde{n})}\frac{\partial}{\partial\Theta_i}\mu(n;\boldsymbol{\Theta})\frac{\partial}{\partial\Theta_j}\mu(\tilde{n};\boldsymbol{\Theta}). \tag{12}$$

The expectation in this last equation appears in the form of a covariance between the random variables $D(n)$ and $D(\tilde{n})$. We assume that pixels in the LADAR image are independent, so the covariance is zero when $n \neq \tilde{n}$. Therefore, each element in the FIM can be reduced to

$$F_{ij} = \sum_n \frac{\frac{\partial}{\partial\Theta_i}\mu(n;\boldsymbol{\Theta})\frac{\partial}{\partial\Theta_j}\mu(n;\boldsymbol{\Theta})}{\sigma^2(n)}. \tag{13}$$

Taking the inverse of this matrix leaves us with a "lower bound" on the covariance matrix for the parameters of interest. By saying that the covariance matrix is "bounded" by the inverse FIM, we mean that $\text{Cov}(\boldsymbol{\Theta}) \geq \mathbf{F}(\boldsymbol{\Theta})^{-1}$ (i.e., the matrix $\text{Cov}(\boldsymbol{\Theta}) - \mathbf{F}(\boldsymbol{\Theta})^{-1}$ is nonnegative definite). This implies that the diagonal elements of the covariance matrix are greater than or equal to the diagonal elements of the inverse FIM. The diagonal elements of the inverse FIM are known as the Cramer-Rao lower bounds for the corresponding variances in the covariance matrix. A number of studies have noted that Cramer-Rao lower bounds are, theoretically, only defined for flat Euclidean spaces. The study performed by Gerwe et al.[4] notes that, in practice, CRLBs can be applied to curved spaces if the bound is relatively small compared to the possible range of values in the space. Since we are concerned with orientation angles of targets, we only have to consider cases where the angle bound is much less than 180 degrees. Note that the maximum error one can have in orientation is 180 degrees.

Another benefit of using Cramer-Rao lower bounds in this manner is that we can adapt them for any type of sensor assuming that the sensor likelihood function between the uncorrupted data values and the measured data values is known. Sensor fusion is naturally incorporated by simply adding the loglikelihoods of the individual sensors.

1.3. Implementation

This analysis was performed on an Apple Macintosh G4 computer running MATLAB 7. Objects were rendered from faceted tank models* using the OpenGL three-dimensional graphics application programming interface

*The models were provided by Dr. Al Curran of the Keeweenaw Research Center, Michigan Technological University. They were designed for the infrared simulation code PRISM, but suffice well for our laser radar study.

(API). Using OpenGL, we perform the transformation $(\tilde{x}, \tilde{y}, \tilde{z}) \rightarrow (\tilde{x}/\tilde{z}, \tilde{y}/\tilde{z})$, taking a three-dimensional scene and turning it into a two-dimensional image through perspective projection. We simulated ideal LADAR data sets by exploiting the depth buffering system employed by OpenGL. The first step is to read the values that OpenGL stores in its depth buffer. Next, these values and the $(x', y')^{\dagger}$ two-dimensional pixel coordinates for the corresponding points in the image can used to undo the perspective projection and obtain the original $(\tilde{x}, \tilde{y}, \tilde{z})$ vectors of three-dimensional world coordinates for the scene. The uncorrupted range values are determined by computing the distance from the camera location to those particular coordinates, producing a range image

To compute the derivatives necessary for determining the CRLBs, a finite difference approximation is employed. Remembering that $\boldsymbol{\mu}$ is a complicated, nonlinear function of the parameter vector $\boldsymbol{\Theta}$ obtained from render($\boldsymbol{\Theta}$), we can apply the following equation:

$$\frac{\partial \mu}{\partial \Theta_i} = \frac{\partial \text{render}(\boldsymbol{\Theta})}{\partial \Theta_i} \approx \frac{\text{render}(\ldots, \Theta_i + \delta, \ldots) - \text{render}(\ldots, \Theta_i - \delta, \ldots)}{2\delta} \tag{14}$$

where δ is some small deviation of the parameter Θ_i, which is a component of $\boldsymbol{\Theta}$, and the ellipses indicate that the remaining parameters are held fixed. The derivative provides us with a representation of how the image changes with small changes of the pose parameters. This technique was originally used to compute derivatives of forward looking infrared (FLIR) loglikelihood functions as part of a diffusion process in an ATR study in Ref. 9.

2. EXPERIMENTS

For the experiments that follow, we assume the laser radar parameters found in Table 1, which were taken from the coherent-detection forward-looking laser radar discussed by Bounds.[10] We assume the laser radar points toward a single M60 tank with a vertical field of view of 12 mrad. The tank parameter space $\boldsymbol{\Theta}$ has variables x, y, and θ, where x and y are horizontal and depth coordinates on the ground, while θ is an angular orientation in degrees, all referenced to some known initial position and orientation. The size of the image obtained from the laser radar is assumed to be 125 x 60 pixels. Images of the tank viewed from the maximum and minimum ranges studied are shown in Figure 1.

Parameter	Value
Optical Efficiency, ϵ_{opt}	0.5
Heterodyne Efficiency, ϵ_{het}	0.5
Detector Quantum Efficiency, η	0.25
Receiver Aperture Dimension, D_R	13 cm
Atmospheric Extinction Coefficient, α	1 dB/km
Average Transmitted Power, P_s	5 W
IF Filter Bandwidth, B	80 MHz
Photon Energy, $h\nu$	1.87×10^{-20} J
Range Resolution, R_{res}	6 m
Target Reflectivity, ρ	0.25

Table 1. Parameter values for the coherent-detection LADAR statistical model.

In the first experiment, we estimated CRLBs for the orientation, horizontal position, and depth position for a target of interest while varying the distance from the laser radar sensor to the target. Each data set was assumed to have a single target sitting on a ground plane with a fixed horizontal position centered on the line of sight of the LADAR sensor. When computing the Cramer-Rao lower bounds, we considered the cases where the parameters are coupled (parameters are estimated jointly) or decoupled (each parameter is estimated individually assuming the other parameters are known). To present bounds for the x and y parameters, we compute the CRLB for

[†]Warning: tildes are used to indicate the three-dimensional world coordinates used when rendering scenes in OpenGL while primes indicate image pixel coordinates.

(a) (b)

Figure 1. Views of the M60 tank at (a) a position close to the laser radar sensor and (b) a position far from the laser radar sensor.

a set of equally-spaced angles through a full 360 degree rotation, and present the *average* of the CRLBs. One could also present similar results for specific rotations of interest.

In the second experiment, we fix the laser radar sensor and the target at a single location. Bounds on the pose parameters are computed at each orientation angle at fixed intervals between 0 and 360 degrees to determine how our estimation ability changes as we view the target from different angles. This experiment only considers the location closest to the LADAR.

3. RESULTS

In the following sections we present charts showing how the bounds on the standard deviation of pose parameters change with orientation angle or distance from target. We note that the absolute values of these bounds are not as significant as the relative trends. The LADAR model that we employ does not account for all possible sources of noise and image corruption, so the bounds are much lower than one would expect them to be in practice.

3.1. CRLBs versus range from LADAR

The Cramer-Rao lower bounds on pose estimation as a function of range from the target of interest behave as one would expect: bounds tend to increase as range increases. This can be attributed to the decrease in target information provided by the sensor (i.e., fewer pixels on target), because as the sensor moves further away the target becomes smaller within the limits of the field of view. The equations for the LADAR statistics also tell us that the local range accuracy increases with range, thus increasing the CRLB even further. This result is consistent for all parameters, as can be seen in Figure 2. Figure 2(d) shows the number of target pixels present in the image at the given ranges from the LADAR.

For larger range values, the curve is not monotonically increasing since small dips appear sporadically along the curve. This may be due to a number of factors, including limitations of the rendering process. Within a small range interval, it is possible that different features of the object being viewed will appear on the final image, and the pixels containing these features may contain different amounts of information. A more theoretical analysis of spatial resolution and sampling may be necessary to determine the exact cause. These non-monotonic qualities of the CRLB curve may also be related to derivative approximations that were necessary when computing the Fisher information matrix. It is clear that the bounds may change when adjusting the derivative stepsizes, but the overall effect of those changes is not entirely clear at present. Some of these issues also arise in work by Gerwe et al.[5]

3.2. CRLBs versus orientation angle

The results of this experiment can be seen in Figure 3. When estimating the bounds on x-position and angle orientation, we see that the CRLBs vary with changes to these parameters, although the variation is difficult to intuitively explain. When estimating y-position, we see noticeable peaks around 90 degrees and 270 degrees. In those instances, the M60 is either pointing toward the LADAR sensor or away from it. This may suggest

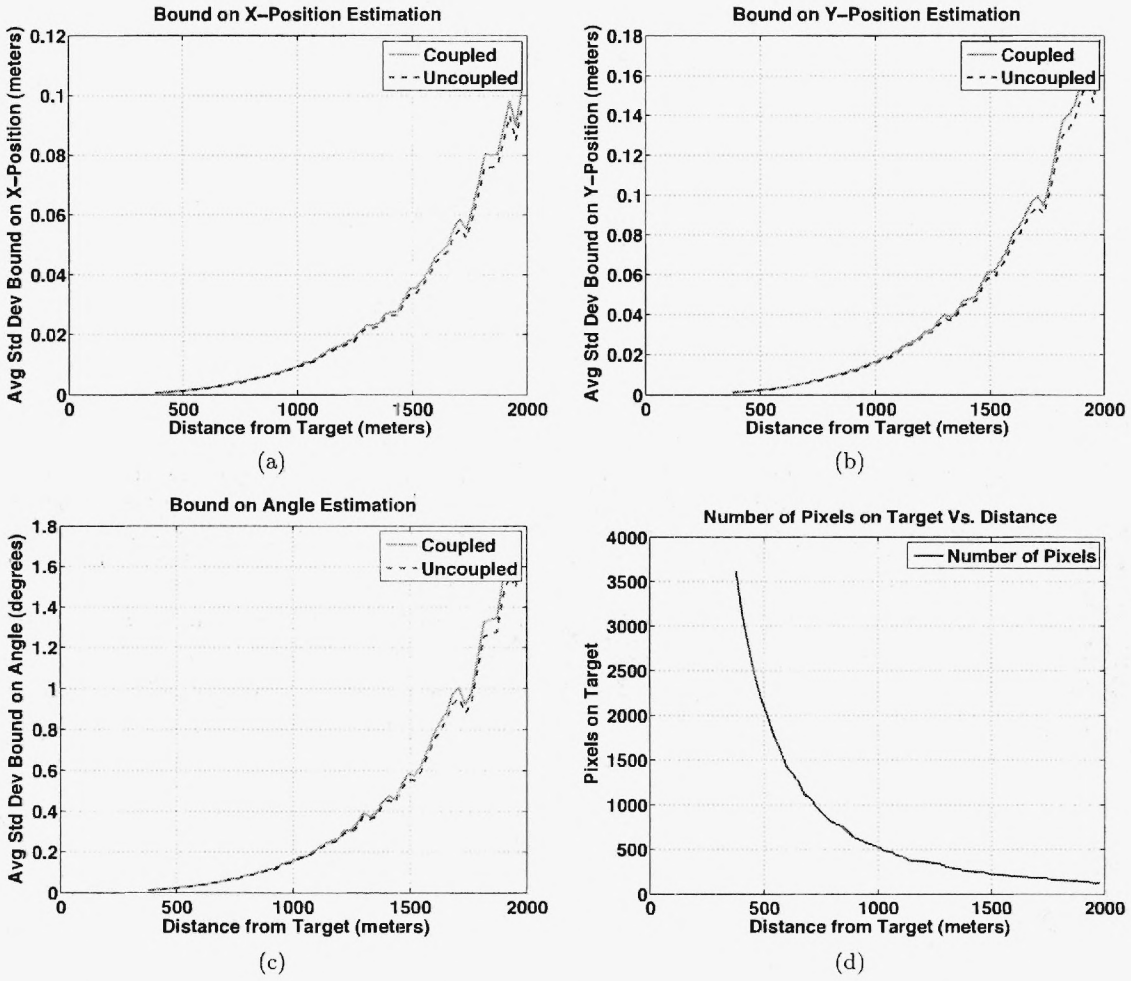


Figure 2. Cramer-Rao lower bounds on estimating (a) x-position, (b) y-position, and (c) orientation angle for a single M60 tank with respect to distance between the LADAR sensor and the tank, and (d) pixels on target versus distance when the M60 is at an orientation angle of 0 degrees (pointing to the right).

that the decrease in visible tank surface area makes it more difficult to estimate the target's depth parameter. Considering that ATR algorithms are typically designed to be invariant to pose, these results suggest that they are indeed affected by it to some degree.

It is interesting to note how the bound changes if parameters are coupled or decoupled. We see, as expected, that the bounds are higher when the parameters must be jointly estimated. In some cases, the bounds are extremely close, as seen with the bounds on y-position and angle estimation in Figures 3(b) and 3(c), respectively. In the case of estimating x-position in Figure 3(a), there are many noticeable gains if the other parameters are known ahead of time. Of course, this will rarely be the case in practice.

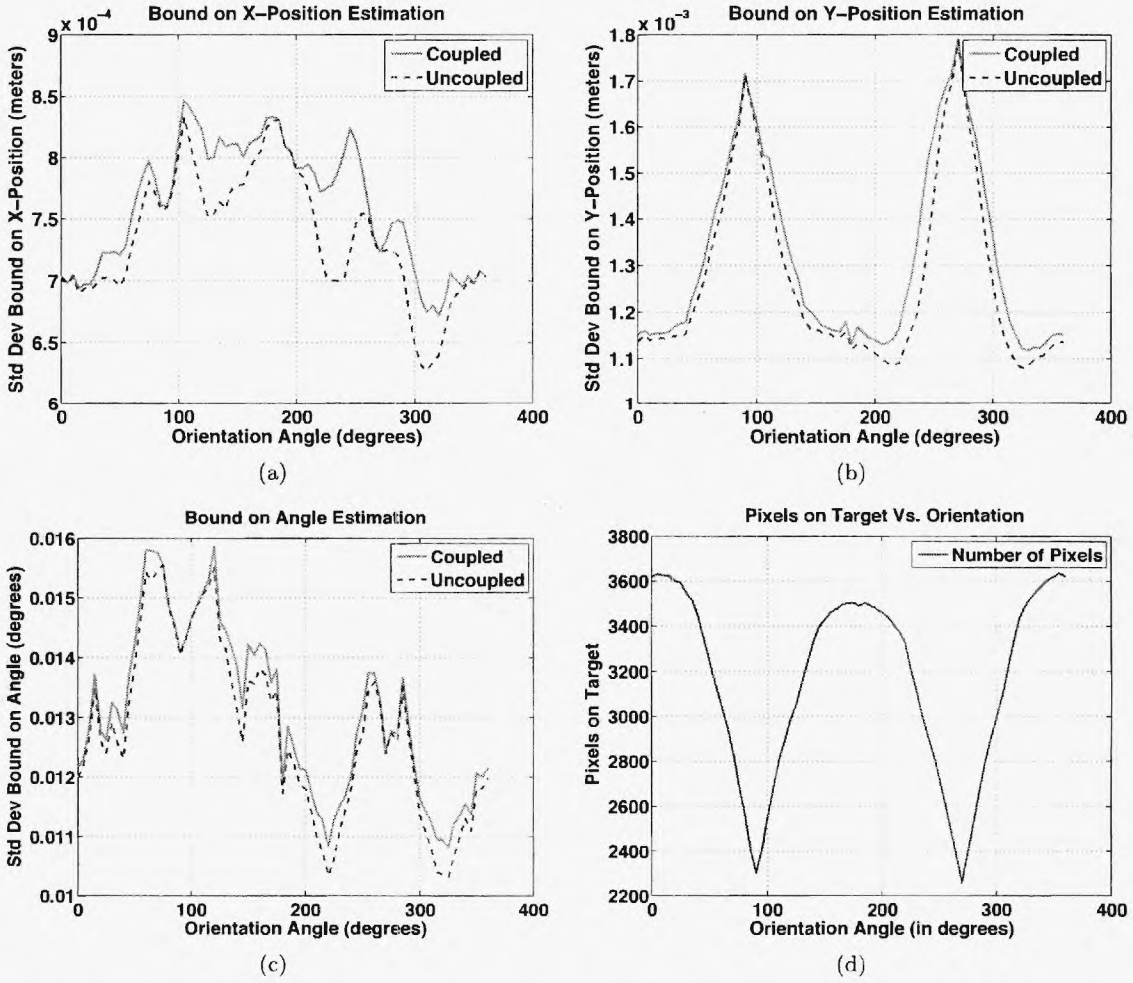


Figure 3. Cramer-Rao bounds on estimating (a) x-position, (b) y-position, and (c) orientation angle for a single M60 tank with respect to the current orientation angle of the tank. Figure (d) shows how the number of pixels on target changes with orientation angle.

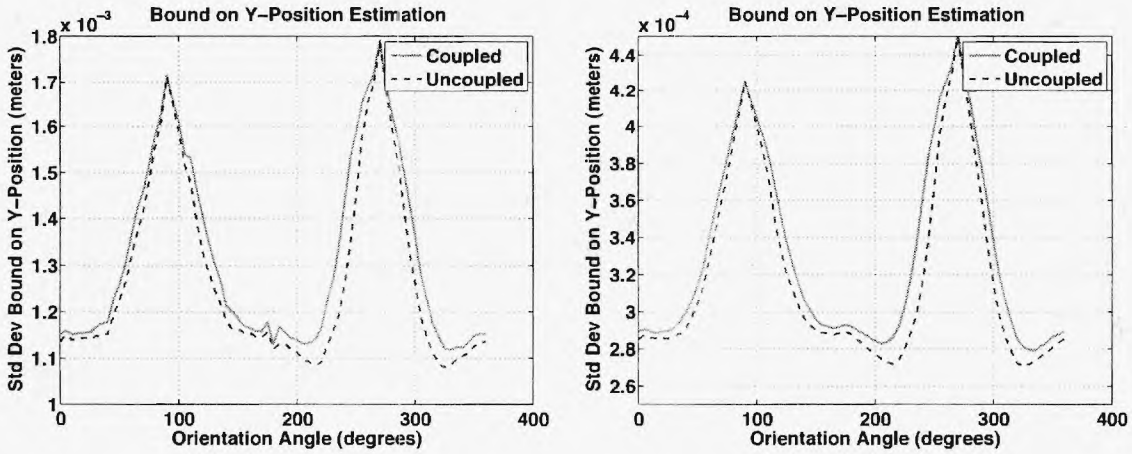
3.3. Image resolution effects

The employed LADAR model acquires range imagery with a resolution of 125×60 pixels. It is interesting to note how the bounds are affected by changes to image resolution. We repeated the experiment defined in the previous section, where we determined the bound on y-position estimation with respect to orientation angle, except this time, we used an image resolution of 500×240 pixels. Sample images from this study are shown in Figure 4.

Plots from the experiment can be seen in Figure 5. The general trends in both plots remain unchanged since there are still strong peaks at the orientation angles where the target is facing directly towards or away from the LADAR sensor. The values of the standard deviations, however, differ by an order of magnitude greater than three. Another major difference between the charts can be seen in the ripples in the CRLB curve for the lower resolution images.

(a) (b)

Figure 4. Images of M60 tanks at different resolutions. Image (a) is 125×60 pixels and image (b) is 500×240 pixels.



(a) (b)

Figure 5. Cramer-Rao bounds on estimating y-position for a single M60 tank with respect to the current orientation angle of the tank. Graph (a) was computed using a resolution of 125×60 pixels and graph (b) was computed using a resolution of 500×240 pixels.

3.4. Choosing derivative stepsizes

One significant problem encountered in this study is the choice of stepsize for the derivatives used to compute the Fisher information matrix. The computation of these derivatives is tricky in that there is no clear analytic means to do so. The plots shown in Figure 6 provide an illustrative example. The stepsizes chosen for the CRLB computations in 6(b) are one quarter the size of those chosen for the computations in 6(a). One noteworthy difference is seen in the absolute magnitudes of the bounds themselves. The bounds computed using the smaller derivative stepsizes are, on average, slightly lower than those computed with the larger derivative stepsizes. A second difference between the two plots is seen in the subtle difference between the coupled and uncoupled curves, specifically that the bounds computed with the larger stepsizes appear to have larger "gaps" between the curves. Intuitively, it makes sense that the smaller stepsizes should offer the more precise derivative computation, but given the nature of the rendering process and the discretization of resulting images, this may not be so. Further

study is needed to determine if ideal stepsizes can be chosen if a better approximation than the first difference may be found.

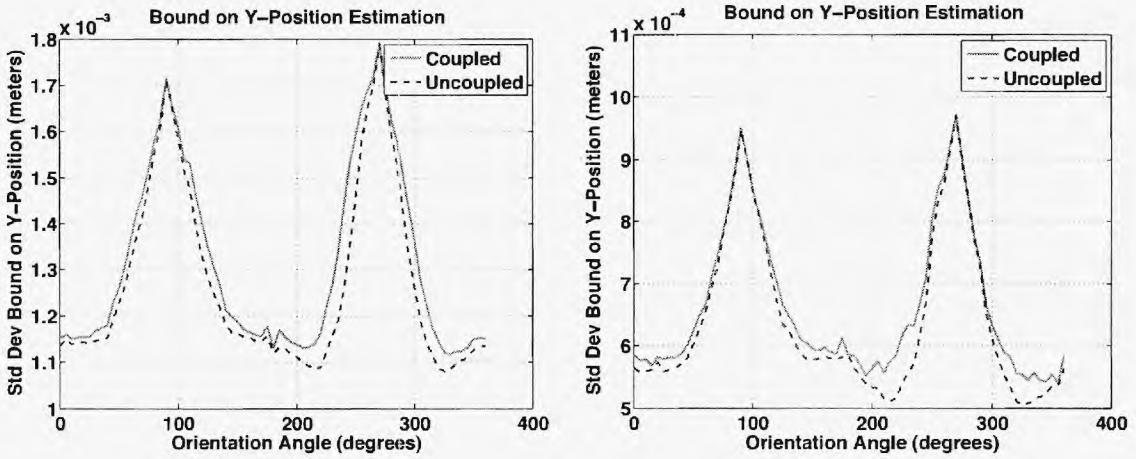


Figure 6. Cramer-Rao bounds on estimating y-position for a single M60 tank with respect to the current orientation angle of the tank. Graph (a) was computed using a derivative stepsize four times the size of that used for the computations in graph (b).

4. CONCLUSION

This paper presented preliminary results on quantifying fundamental limits on target pose estimation performance for laser radar imagery. We considered Cramer-Rao lower bounds on the variance of unbiased estimators of pose parameters of interest. We have shown that the ability to estimate pose parameters appears to depend heavily on true pose in that different features can be seen from the laser radar when viewing targets from different positions and orientations. A good feature of this procedure is that it can be performed for any type of sensor as long as the sensor/data likelihood function is known and the CRLB can be derived. A more theoretical analysis on computing the numerical derivatives needed by the Fisher information matrix, artifacts introduced when rendering laser radar imagery, and the effects caused by the finite spatial resolution of the images may be necessary to make this technique more practical. Ideally, an ATR system designer should be able to input sensor parameters and target modes and determine the bounds on the standard deviation of parameter estimators.

The CRLBs can give target recognition algorithm developers a goal to shoot for. If performance is already near the bound, then there may be little point in spending more resources to further improving the algorithm. In particular, if the performance does not match the needs of the user, the bounds may tell us that further effort on algorithm development will be wasted; a better - namely, a more informative - sensor is needed. If the performance matches the needs of the user, but is not near the bound, then that suggests that similar performance might be achieved using a more sophisticated algorithm in conjunction with a less expensive sensor. Such trade-offs are important to analyze, particularly since the cost of computing hardware tends to follow Moore's law, while the cost of sensors remains relatively fixed.

This paper has solely considered the bounds on estimating pose parameters assuming the target type is known. Our future work will also consider bounds on the performance of target recognition algorithms. One important result arising from the work of Grenander, Miller, and Srivastava[11] is that recognition performance is inherently linked to the ability to estimate nuisance parameters such as the orientation and location of a target. Even if a particular algorithm is designed to be invariant to pose and does not explicitly estimate pose parameters, the issue percolates under the surface, affecting how well the algorithm can perform.

ACKNOWLEDGMENTS

This work was sponsored by the Air Force Office of Scientific Research (AFOSR) grant F49620-03-1-0340 .

REFERENCES

1. A. E. Koksal, J. H. Shapiro, and M. I. Miller, "Performance analysis for ground-based target orientation estimation: FLIR/LADAR sensor fusion," *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers* **vol.2**, pp. 1240–4, (Pacific Grove, CA), 1999.
2. U. Grenander, M. I. Miller, and A. Srivastava, "Hilbert-Schmidt lower bounds for estimators on matrix lie groups for ATR," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(8), pp. 790–802, 1998.
3. A. Jain, P. Moulin, M. I. Miller, and K. Ramchandran, "Information-theoretic bounds on target recognition performance based on degraded image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(9), pp. 1153–66, 2002.
4. D. R. Gerwe, J. L. Hill, and P. S. Idell, "Cramer-Rao analysis of orientation estimation: Influence of target model uncertainties," *Journal of the Optical Society of America A: Optics and Image Science, and Vision* **20**(5), pp. 817–826, 2003.
5. D. R. Gerwe and P. S. Idell, "Cramer-Rao analysis of orientation estimation: Viewing geometry influences on the information conveyed by target features," *Journal of the Optical Society of America A: Optics and Image Science, and Vision* **20**(5), pp. 797–816, 2003.
6. J. Green, T. J. and J. H. Shapiro, "Maximum-likelihood laser radar range profiling with the expectation-maximization algorithm," *Optical Engineering* **31**(11), pp. 2343–54, 1992.
7. J. Green, T. J. and J. H. Shapiro, "Detecting objects in three-dimensional laser radar range images," *Optical Engineering* **33**(3), pp. 865–74, 1994.
8. A. D. Lanterman, "Jump-diffusion algorithm for multiple target recognition using laser radar range data," *Optical Engineering* **40**(8), pp. 1724–1728, 2001.
9. A. D. Lanterman, M. I. Miller, and D. L. Snyder, "General Metropolis-Hastings jump diffusions for automatic target recognition in infrared scenes," *Optical Engineering* **36**(4), pp. 1123–37, 1997.
10. J. K. Bounds, "The infrared airborne radar sensor suite," Tech. Rep. RLE Technical Report No. 610, Massachusetts Institute of Technology, December 1996.
11. U. Grenander, A. Srivastava, and M. I. Miller, "Asymptotic performance analysis of Bayesian target recognition," *IEEE Transactions on Information Theory* **46**(4), pp. 1658–65, 2000.

Appendix D

L.M. Ehrman and A.D. Lanterman, "Chernoff-Based Prediction of ATR Performance from Rician Radar Data, with Application to Passive Radar," *Optical Engineering*, letter of acceptance subject to minor revision received Feb. 2, 2006; revision submitted Feb. 2007.

Chernoff-Based Prediction of ATR Performance from Rician Radar Data, with Application to Passive Radar

Lisa M. Ehrman and Aaron D. Lanterman

Center for Signal and Image Processing
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA

Telephone: 770-528-7079, 404-385-2548

I. Abstract

This paper develops a method for quickly assessing the performance of an ATR algorithm without using computationally-expensive Monte Carlo trials. To do so, it exploits the relationship between the probability of error in a binary hypothesis test under the Bayesian framework to the Chernoff information. Since it has been demonstrated in prior work that the RCS profiles being used to identify the targets are well-modeled as Rician, we begin by deriving a closed-form approximation for the Chernoff information between two Rician densities. This leads to an approximation for the probability of error in the ATR algorithm that is a function of the number of measurements available. We conclude the paper with an application that would be particularly cumbersome to accomplish via Monte Carlo trials, but that can be quickly addressed using the Chernoff information approach. This application evaluates the length of time that an aircraft must be tracked before the probability of error in the ATR algorithm drops below a desired threshold.

II. RCS-Based Target Identification

A. Past Approaches to ATR

The literature surrounding the recognition of fast-moving fixed-wing aircraft is typically divided into two schools of thought. On one side of the debate are researchers who propose the creation of target images to accomplish identification. Advocates of this approach suggest everything from two-dimensional inverse synthetic aperture radar (ISAR) images to a sequence of one-dimensional range profiles [1]. The alternate approach bypasses the creation of images and attempts recognition directly on the data. Herman [2], [3] takes this second approach to automatic target recognition (ATR), using data obtained from a passive radar system.

Although ATR has been a subject of much research, Herman's application of passive radar was innovative. Unlike traditional radar systems, passive radar systems bypass the need for dedicated transmitters by exploiting "illuminators of opportunity" such as commercial television and FM radio signals. In doing so, they are able to reap a number of benefits. Most notably, the fact that passive radar systems do not emit energy renders them covert. An additional benefit is that the illuminators of opportunity often operate at

much lower frequencies than their traditional counterparts. It has long been proposed that low-frequency signals are well-suited for ATR [4], [5], [6]. Several passive radar systems have been developed in recent years, with Lockheed Martins' Silent Sentry and John Sahr's Manastash Ridge Radar [7], [8] serving as notable examples.

B. Coupling Passive Radar RCS-Based Target Recognition with a Coordinated Flight Model

The goal of this research is to enhance existing passive radar systems, which are already capable of detecting and tracking aircraft, with target recognition capabilities. In particular, the proposed approach to target recognition, which falls into the second school of thought on ATR, uses the covertly obtained RCS from a passive radar source as the primary parameter for identification. More precisely, target recognition is conducted by comparing the covertly collected RCS of the aircraft under track to the simulated RCS of a variety of aircraft comprising the target library.

To make the simulated RCS as accurate as possible, the received signal model accounts for aircraft position and orientation, propagation losses, and antenna gain patterns. A coordinated flight model uses the target state vector produced by the passive radar tracker to approximate aircraft orientation. Coupling the aircraft orientation and state with the known antenna locations renders computation of the incident and observed azimuth and elevation angles possible. The Fast Illinois Solver Code (FISC) simulates the RCS of potential target classes as a function of these angles. Thus, the approximated incident and observed angles allow the appropriate RCS to be extracted from a database of FISC results. Using this process, the RCS of each aircraft in the target class is simulated as though each is executing the same maneuver as the target detected by the system. Two additional scaling processes are required to transform the RCS into a power profile simulating the signal arriving at the receiver. First, the RCS is scaled by the Advanced Refractive Effects Prediction System (AREPS) code to account for propagation losses that occur as functions of altitude and range. Then, the Numerical Electromagnetic Code (NEC2) computes the antenna gain pattern, further scaling the RCS. A Rician likelihood model compares the scaled RCS of the illuminated aircraft with those of the potential targets, resulting in identification. The result is an algorithm for covertly identifying aircraft with a low-cost passive radar system.

III. Assessing the Performance of the ATR Algorithm Via the Chernoff Information

Prior work [9], [10], [11] has shown that the proposed ATR algorithm has merit. However, to more fully test the algorithm against aircraft in a variety of locations executing a variety of maneuvers, a staggering number of Monte Carlo trials are required. To combat this dilemma, a more efficient approach for assessing the ATR algorithm's capabilities is desired. This paper describes one such approach.

Under the Bayesian framework, the probability of error in a binary hypothesis testing problem can be approximated as a function of the Chernoff information between two densities. In the case of this ATR algorithm, the densities representing the magnitude of the RCS of two aircraft being compared are best modeled as Rician. For this reason, Section III-B derives a closed-form approximation for the Chernoff information between two Rician densities. Section IV then compares the ATR performance predicted by the Chernoff information approach with that obtained using Monte Carlo trials. Having shown that the two approaches provide similar results, Section V then uses the Chernoff information to determine how long an aircraft must be tracked in order for the probability of error in the ATR algorithm to drop below a desired threshold. This application demonstrates the advantage of using the Chernoff information to approximate the performance of the ATR algorithm. Since the Chernoff information is a function of the number of measurements collected, its application to this problem is quite natural. Monte Carlo trials, in contrast, would make for a particularly cumbersome approach to the problem.

A. Approximating the Probability of Error Via the Chernoff Information

It is widely known that the probability of error in a binary hypothesis test, conducted in the Bayesian framework, can be approximated in terms of the Chernoff information [12]. In particular, the probability of error is approximated with

$$P_E \approx e^{-C(p(x),q(x))}, \tag{1}$$

where $C\left(p(x), q(x)\right)$ is the Chernoff information. The Chernoff information between two densities, $p(x)$ and $q(x)$ is typically found using

$$C\left(p(x), q(x)\right) = -min_{0 \leq \lambda \leq 1}\left\{\mu(\lambda)\right\}, \tag{2}$$

where $\mu(\lambda)$ is given by

$$\mu(\lambda) = \ln\left[\int_x q(x)^{\lambda} p(x)^{1-\lambda} dx\right]. \tag{3}$$

To approximate the probability of error in the proposed ATR algorithm between two aircraft, $p(x)$ and $q(x)$ are set to the Rician densities [3]

$$p(x) = \frac{x}{\sigma^2} e^{\frac{-(x^2+s_p^2)}{2\sigma^2}} I_0\left[\frac{x s_p}{\sigma^2}\right] \tag{4}$$

and

$$q(x) = \frac{x}{\sigma^2} e^{\frac{-(x^2+s_q^2)}{2\sigma^2}} I_0\left[\frac{x s_q}{\sigma^2}\right], \tag{5}$$

where I_0 is the zeroth-order modified Bessel function of the first kind. Note that x is the magnitude of the covertly collected RCS of the aircraft being tracked, s_p and s_q are the magnitudes of the simulated RCS for both aircraft, and σ^2 is the noise power.

Substituting [4] and [5] into [3] results in

$$\mu(\lambda) = \ln\left\{\int_0^{\infty}\left[\frac{x}{\sigma^2} e^{\frac{-(x^2+s_q^2)}{2\sigma^2}} I_0\left(\frac{x s_q}{\sigma^2}\right)\right]^{\lambda}\left[\frac{x}{\sigma^2} e^{\frac{-(x^2+s_p^2)}{2\sigma^2}} I_0\left(\frac{x s_p}{\sigma^2}\right)\right]^{1-\lambda} dx\right\}. \tag{6}$$

Rearranging [6] results in

$$\mu(\lambda) = \ln\left\{\int_0^{\infty}\left[\frac{\frac{x}{\sigma^2} e^{\frac{-(x^2+s_q^2)}{2\sigma^2}} I_0\left(\frac{x s_q}{\sigma^2}\right)}{\frac{x}{\sigma^2} e^{\frac{-(x^2+s_p^2)}{2\sigma^2}} I_0\left(\frac{x s_p}{\sigma^2}\right)}\right]^{\lambda} \frac{x}{\sigma^2} e^{\frac{-(x^2+s_p^2)}{2\sigma^2}} I_0\left(\frac{x s_p}{\sigma^2}\right) dx\right\}, \tag{7}$$

which is further reduced to

$$\mu(\lambda) = \ln\left\{\int_0^{\infty} e^{\frac{\lambda(s_p^2-s_q^2)}{2\sigma^2}} \frac{x}{\sigma^2} e^{\frac{-(x^2+s_p^2)}{2\sigma^2}} I_0\left(\frac{x s_p}{\sigma^2}\right)\left[\frac{I_0\left(\frac{x s_q}{\sigma^2}\right)}{I_0\left(\frac{x s_p}{\sigma^2}\right)}\right]^{\lambda} dx\right\}. \tag{8}$$

B. Deriving a Closed-Form Approximation for the Chernoff Information

The presence of Bessel functions in [8] render computation of an analytical solution quite difficult. However, this can be accomplished by applying the Laplace Method to the integral. The first exponential term is not a function of x, so it is pulled outside the integral. Applying the Laplace Method to essentially just rewrites the remaining terms from [8] in $e^{ln()}$ form, or

$$\mu(\lambda) = \ln\left\{e^{\frac{\lambda(s_p^2-s_q^2)}{2\sigma^2}}\int_0^\infty e^{ln\left(\frac{x}{\sigma^2}\right)-\frac{(x^2+s_p^2)}{2\sigma^2}-(\lambda-1)ln\left[I_0\left(\frac{xs_p}{\sigma^2}\right)\right]+\lambda ln\left[I_0\left(\frac{xs_q}{\sigma^2}\right)\right]}dx\right\}. \tag{9}$$

This is equivalent to

$$\mu(\lambda) = ln\left[e^{\frac{\lambda(s_p^2-s_q^2)}{2\sigma^2}}\int_0^\infty e^{h(x,\lambda)}dx\right], \tag{10}$$

where

$$h(x,\lambda) = ln\left(\frac{x}{\sigma^2}\right) - \frac{(x^2+s_p^2)}{2\sigma^2} - (\lambda-1)ln\left[I_0\left(\frac{xs_p}{\sigma^2}\right)\right] + \lambda ln\left[I_0\left(\frac{xs_q}{\sigma^2}\right)\right]. \tag{11}$$

Thus, $\mu(\lambda)$ reduces to

$$\mu(\lambda) \approx \frac{\lambda(s_p^2-s_q^2)}{2\sigma^2} + h(\hat{x},\lambda) + \frac{1}{2}ln\left(\frac{-2\pi}{h''(\hat{x},\lambda)}\right), \tag{12}$$

where $\hat{x}$ is the value of x found by setting the derivative of $h(x,\lambda)$ equal to zero. To make the math tractable, the (zeroth order and first order) Bessel functions are approximated as $I_0(y) \approx \exp(y)$ and $I_1(y) \approx \exp(y)$. This works best if the Bessel function arguments are large, as is likely to be the case in practice. This approximation results in two solutions. However, it is trivial to show that, given the limits of integration of the Rician density, the only valid solution is

$$\hat{x} = \frac{1}{2}\lambda(s_q - s_p) + \frac{1}{2}s_p + \frac{1}{2}\sqrt{\lambda^2(s_q^2+s_p^2-2s_ps_q) + 2\lambda s_ps_q - 2\lambda s_p^2 + s_p^2 + 4\sigma^2}. \tag{13}$$

The second derivative of $h(x,\lambda)$ is given by

$$h''(x,\lambda) = -\frac{1}{\sigma^2} - \frac{1}{x^2} - \left(\frac{\lambda s_q^2}{\sigma^4}\right)\left[\frac{I_1^2\left(\frac{xs_q}{\sigma^2}\right)}{I_0^2\left(\frac{xs_q}{\sigma^2}\right)}\right] + \left(\frac{\lambda s_q^2}{2\sigma^4}\right)\left[\frac{I_0\left(\frac{xs_q}{\sigma^2}\right)+I_2\left(\frac{xs_q}{\sigma^2}\right)}{I_0\left(\frac{xs_q}{\sigma^2}\right)}\right]\ldots. \tag{14}$$

$$\ldots - \left(\frac{(\lambda-1)s_p^2}{\sigma^4}\right)\left[\frac{I_1^2\left(\frac{xs_p}{\sigma^2}\right)}{I_0^2\left(\frac{xs_p}{\sigma^2}\right)}\right] + \left(\frac{(\lambda-1)s_p^2}{2\sigma^4}\right)\left[\frac{I_0\left(\frac{xs_p}{\sigma^2}\right) + I_2\left(\frac{xs_p}{\sigma^2}\right)}{I_0\left(\frac{xs_p}{\sigma^2}\right)}\right]. \quad (15)$$

Thus, the probability of error in a binary Bayesian hypothesis test is approximated by

$$P_E \approx e^{-min_{0\leq\lambda\leq 1}\left[\frac{\lambda(s_p^2-s_q^2)}{2\sigma^2}+h(\hat{x},\lambda)+\frac{1}{2}ln\left(\frac{-2\pi}{h''(\hat{x},\lambda)}\right)\right]}. \quad (16)$$

IV. Comparison of Chernoff Information Predictions with Monte Carlo Results

To demonstrate that the Chernoff information can be used to approximate the probability of error of the ATR algorithm with a reasonable degree of success, its predictions for the probability of error are compared to those obtained through Monte Carlo trials. Three trajectories are used in the comparison. The first two trajectories involve aircraft flying in straight-and-level paths with velocities of 200 m/s and altitudes of 8000 m. In the first straight-and-level trajectory, the aircraft fly directly away from the receiver, while in the second, they fly broadside to it. Because a much broader range of aspect angles are visible to the receiver in the second flight path, the probability of error is expected to be lower than in the first. Finally, the third trajectory is a constant-altitude banked turn in which the aircraft velocity is 100 m/s and the altitude is 8000 m.

To thoroughly compare the probability of error predicted using Chernoff information and that computed using Monte Carlo trials, each is computed for a number of noise figures [1]. In particular, the simulations are conducted with the noise figure varying from 30 to 100 dB in increments of 5 dB. Note that this extends to noise figures much larger than would ever be expected in a real system, simply so that the breaking point of the algorithm is visible. Prior work demonstrates that the maximum noise figure ever anticipated in the proposed system is 45 dB.

Figure VII shows the probability of error obtained using the first straight-and-level flight path, as a function of the noise figure, computed by both approaches. Two points are worth making. First, both methods agree that the probability of error is near zero

[1]The noise figure is a unitless quantity that increases proportionately to the noise power. It should not be confused with SNR. For more information, see the papers listed in the bibliography by Ehrman and Lanterman.

until the noise figure reaches 50 dB. This noise level is higher than the maximum noise level anticipated in any real system; thus, the algorithm is expected to perform very well at realistic noise levels. Second, the results obtained using the Chernoff information corroborate those obtained using Monte Carlo trials. Although there is slight disagreement during the transition period, with noise figures between 50 and 60 dB, the results obtained using the Chernoff information generally agree with those obtained from Monte Carlo trials.

Figure 2 shows the probability of error curves for the second straight-and-level trajectory. Although the algorithm's performance has improved, now that a wider range of aspects are presented to the receiver, the results obtained using the Chernoff information still corroborate those obtained using Monte Carlo trials. The trend continues using the banked-turn trajectory, whose probability of error curves are given in Figure 3.

V. Using the Chernoff Information to Efficiently Analyze Performance of an ATR Algorithm

Section IV demonstrates that the approximation of the algorithm's performance using the Chernoff information is very similar to that which is obtained using Monte Carlo trials. As such, the Chernoff information approach can confidently be used to address questions that would be cumbersome to address via Monte Carlo trials. For example, a useful piece of information is the length of time that the aircraft must be tracked in order to identify it with a desired probability of error. The number of Monte Carlo trials required to address this question is staggering, as a complete set of trials would be required for each period of time tested. However, this problem is easily addressed using the Chernoff information.

Figures 4 and 5 show the probability of error as a function of the length of time that the target is tracked, using noise figures of 40 and 45 dB, for the first straight-and-level trajectory. Similar results are given in Figures 6 through 9 for the second straight-and-level trajectory and the banked-turn trajectory, respectively.

Several points are worth stating. Consider the first straight-and-level trajectory. Whether the noise figure is 40 or 45 dB, the most difficult comparisons for the ATR algorithm are the between the F-15 and T-38, and the Falcon-20 and Falcon-100. This corroborates fairly well with the results presented in Figure VII, and makes intuitive sense, since the

F-15 and T-38 are both fighter-style aircraft while the Falcon-20 and Falcon-100 are both commercial. Another noteworthy point is that the probability of a correct identification when considering the Falcon-20 and Falcon-100 never exceeds 70% if the noise figure is 45 dB and the maximum time spent collecting data is 50 seconds. This is largely attributed to the scenario. Since the aircraft fly directly away from the receiver, the range of aspects presented to the receiver is very narrow. Better performance is expected using the second straight-and-level maneuver, in which the aircraft fly broadside to the receiver.

This is indeed the case. Using the second straight-and-level trajectory with a noise figure of 40 dB, the ATR algorithm can correctly distinguish between all pairs of aircraft with a probability of error below 5% within 12 seconds, instead of nearly 50. Similarly, when the noise figure increases to 45 dB, the algorithm correctly identify all pairs of aircraft within 14 seconds. When using the first straight-and-level maneuver, the algorithm was never able to reach this level of certainty. Clearly, the ATR algorithm's performance improves as a broader range of aspect angles are presented to the receiver.

The ATR algorithm performs even better against aircraft executing the banked turn maneuver. This is probably caused by the same trend just described. The more aspects of the aircraft presented to the receiver, the better the ATR algorithm performs. Using the second straight-and-level maneuver, it takes 12-14 seconds, (for noise figures of 40-45 dB), for enough aspects to be presented to the receiver that the probability of error drops below 5%. Since aircraft are constantly presenting new aspects to the receiver when executing the banked turn maneuver, the probability of error decreases even more rapidly, with the exception of the Falcon-20 and Falcon-100 pair. In this case, the aircraft look very similar to each other at the aspects angles initially presented to the receiver.

VI. Conclusions

Through the development of the closed-form approximation of the Chernoff information between two Rician densities, and its application to the probability of error in a binary hypothesis testing problem under the Bayesian framework, this paper develops a means for rapidly assessing the performance of a covert target recognition algorithm. Monte Carlo trials have already suggested that the ATR algorithm will be successful at the anticipated noise levels. The new approach for assessing the algorithm's performance allows it to be

more thoroughly tested. Evaluating the ATR algorithm using real (rather than simulated) data is reserved for future work.

REFERENCES

[1] S. Jacobs and J. O'Sullivan, "Automatic target recognition using sequences of high resolution radar range-profiles," *IEEE Trans. on Aerospace and Electronic Systems* **36**(2), pp. 364–382, 2000.

[2] S. Herman, *A Particle Filtering Approach to Joint Passive Radar Tracking and Target Classification*, Doctoral Dissertation, Department of Electrical and Computer Engineering, Univ. of Illinois at Urbana-Champaign, Urbana, IL, 2002.

[3] S. Herman and P. Moulin, "A particle filtering approach to joint radar tracking and automatic target recognition," in *Proc. IEEE Aerospace Conference*, (Big Sky, Montana), March 10-15 2002.

[4] Y. Lin and A. Ksienski, "Identification of complex geomtrical shapes by means of low-frequency radar returns," *The Radio and Electronic Engineer* **46**, pp. 472–486, Oct. 1976.

[5] H. Lin and A. Ksienski, "Optimum frequencies for aircraft classification," *IEEE Trans. on Aerospace and Electronic Systems* **17**, pp. 656–665, Sept. 1981.

[6] J. Chen and E. Walton, "Comparison of two target classification techniques," *IEEE Trans. on Aerospace and Electronic Systems* **22**, pp. 15–21, Jan. 1986.

[7] J. Sahr and F. Lind, "The Manastash ridge radar: A passive bistatic radar for upper atmospheric radio science," *Radio Science* , pp. 2345–2358, Nov.-Dec. 1997.

[8] J. Sahr and F. Lind, "Passive radio remote sensing of the atmosphere using transmitters of opportunity," *Radio Science* , pp. 4–7, March 1998.

[9] L. Ehrman and A. Lanterman, "Automated target recogntion using passive radar and coordinated flight models," in *Automatic Target Recognition XIII*, **SPIE Proc. 5094**, (Orlando, FL), April 2003.

[10] L. Ehrman and A. Lanterman, "Target identification using modeled radar cross sections and a coordinated flight model," in *Proceedings from the Third Multi-National Conference on Passive and Covert Radar*, (Seattle, WA), October 2003.

[11] L. Ehrman and A. Lanterman, "A robust algorithm for automated target recognition using passive radar," in *Proceedings from the IEEE Southeastern Symposium on System Theory*, (Atlanta, GA), March 2004.

[12] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc., 1991.

VII. Biographies

Lisa M. Ehrman received a B.S. in electrical engineering from the University of Dayton in May of 2000. She worked for the following two years for MacAulay Brown, Inc. supporting the Suite of Integrated Infrared Countermeasures (SIIRCM) program lead by the United States Army. Her main role was in the Test and Evaluation group, but she also supported the Modeling and Simulation side of the program. In pursuit of a research-oriented career, Lisa left MacAulay Brown in 2002 and began attending the Georgia Institute of Technology. She received her M.S. in electrical and computer engineering in May 2004, and her Ph.D. in electrical and computer engineering in December 2005. Her Ph.D. dissertation focused on automatic target recognition via passive radar and an EKF for estimating aircraft orientation. She has been a full-time research engineer at the Georgia Tech Research Institute (GTRI) since May 2004. Her work at GTRI has included comparing and developing tracking algoirthms for ballistic missile defense, feature-assisted tracking, tracking unresolved separating targets using monopulse radar, and the development of launch point estimation and impact point prediction algorithms for small ballistic targets.

Aaron D. Lanterman is an Assistant Professor of Electrical and Computer ¿Engineering at the Georgia Institute of Technology, which he joined in the fall of 2001. In 2004, he was chosen to hold the Demetrius T. Paris Professorship, a special chaired position for the development of young faculty. He finished a triple major consisting of a B.A. in Music, B.S. in Computer Science, and B.S. in Electrical Engineering at Washington University in St. Louis in 1993. He stayed on for graduate school, receiving an M.S. (1995) and D.Sc. (1998) in Electrical Engineering. His graduate work focused on target recognition for infrared imagery as part of the multi-university U.S. Army Center for Imaging Science. After graduation, he joined the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign as a postdoctoral research associate and then as a visiting assi stant professor, where he managed a large project on covert radar systems exploiting illuminators of opportunity such as television and FM radio signals. His other research interests include target tracking, image reconstruction, and music synthesis. In 2006, he received the Richard M. Bass Outstanding Teacher Award, as voted on by the senior class of the School of Electrical and Computer Engineering at Georgia Tech.

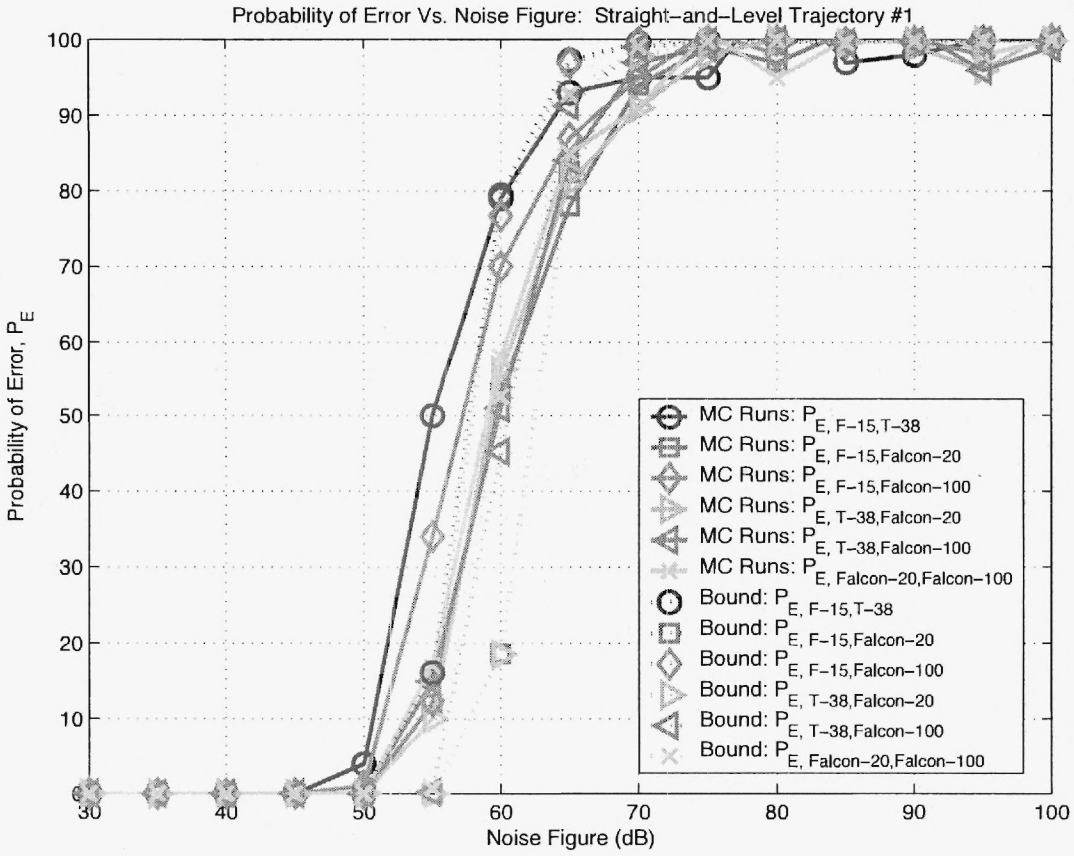


Fig. 1. Probability of Error Vs. Noise Figure: Straight-and-Level Trajectory #1

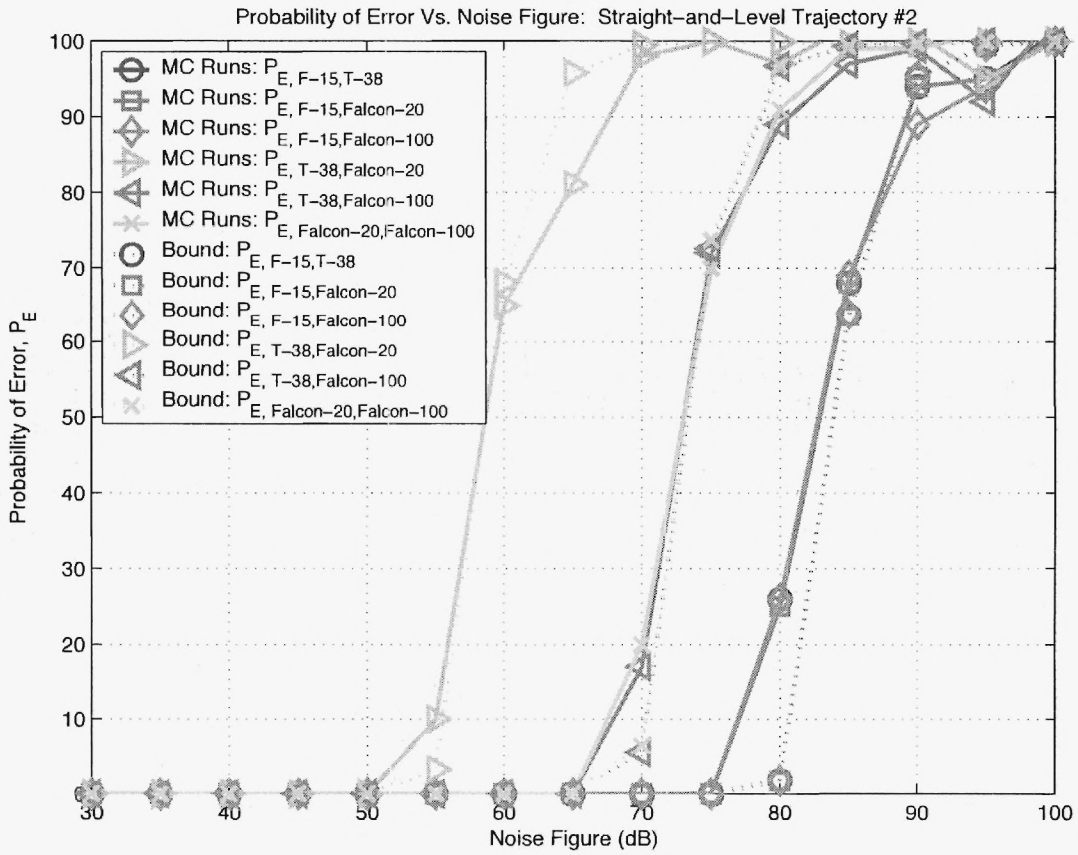


Fig. 2. Probability of Error Vs. Noise Figure: Straight-and-Level Trajectory #2

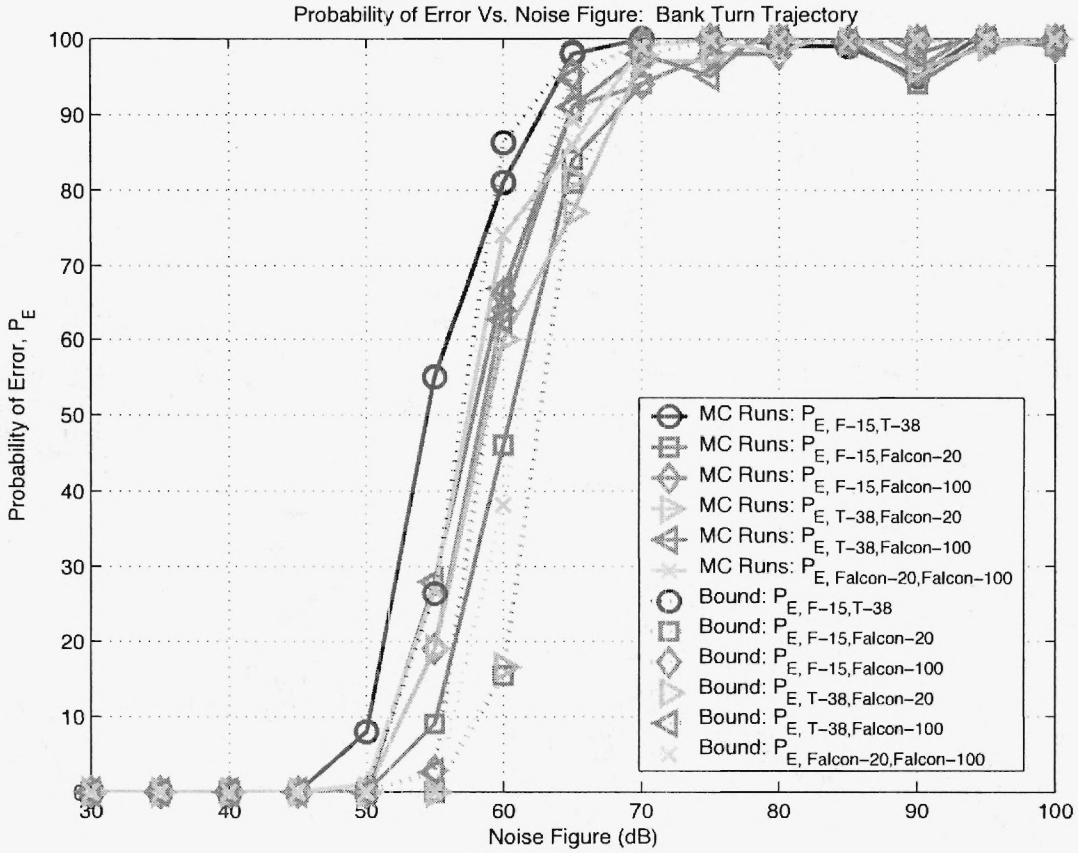


Fig. 3. Probability of Error Vs. Noise Figure: Banked Turn Trajectory

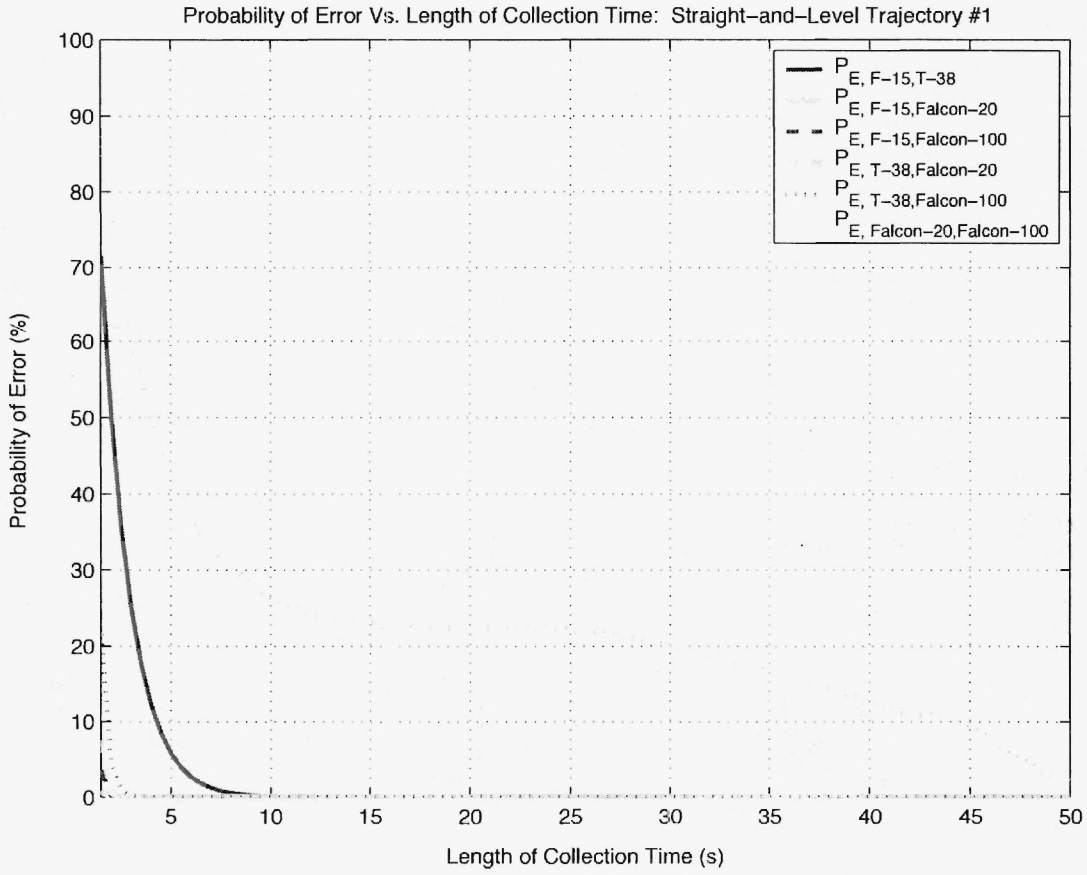


Fig. 4. Probability of Error Vs. Time: Straight-and-Level Trajectory #1, Noise Figure = 40 dB

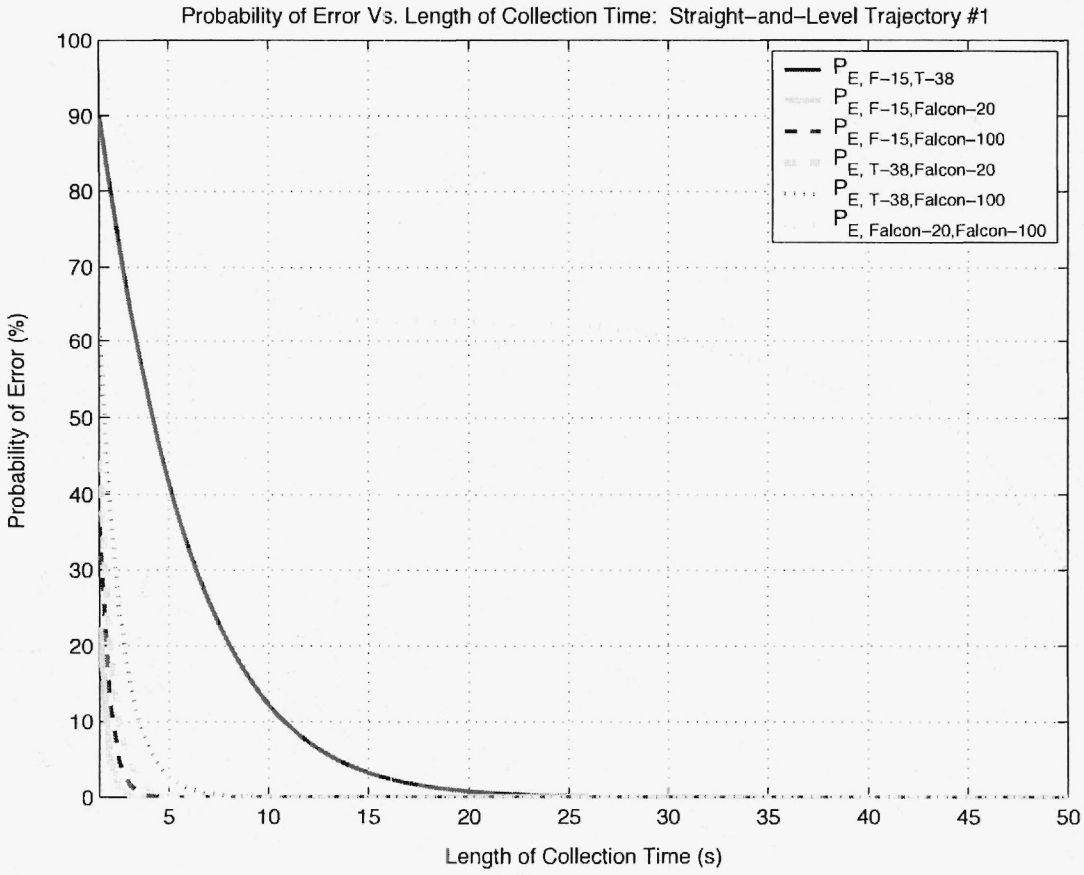


Fig. 5. Probability of Error Vs. Time: Straight-and-Level Trajectory #1, Noise Figure = 45 dB

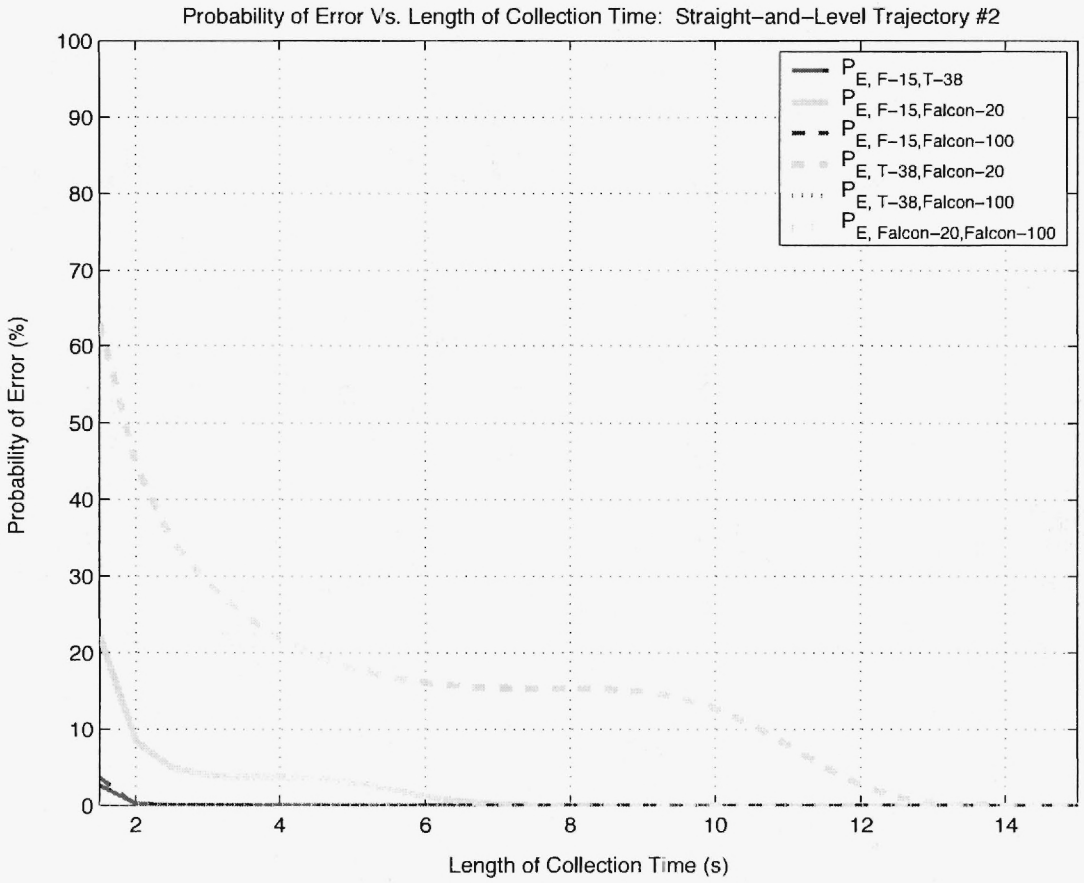


Fig. 6. Probability of Error Vs. Time: Straight-and-Level Trajectory #2, Noise Figure = 40 dB

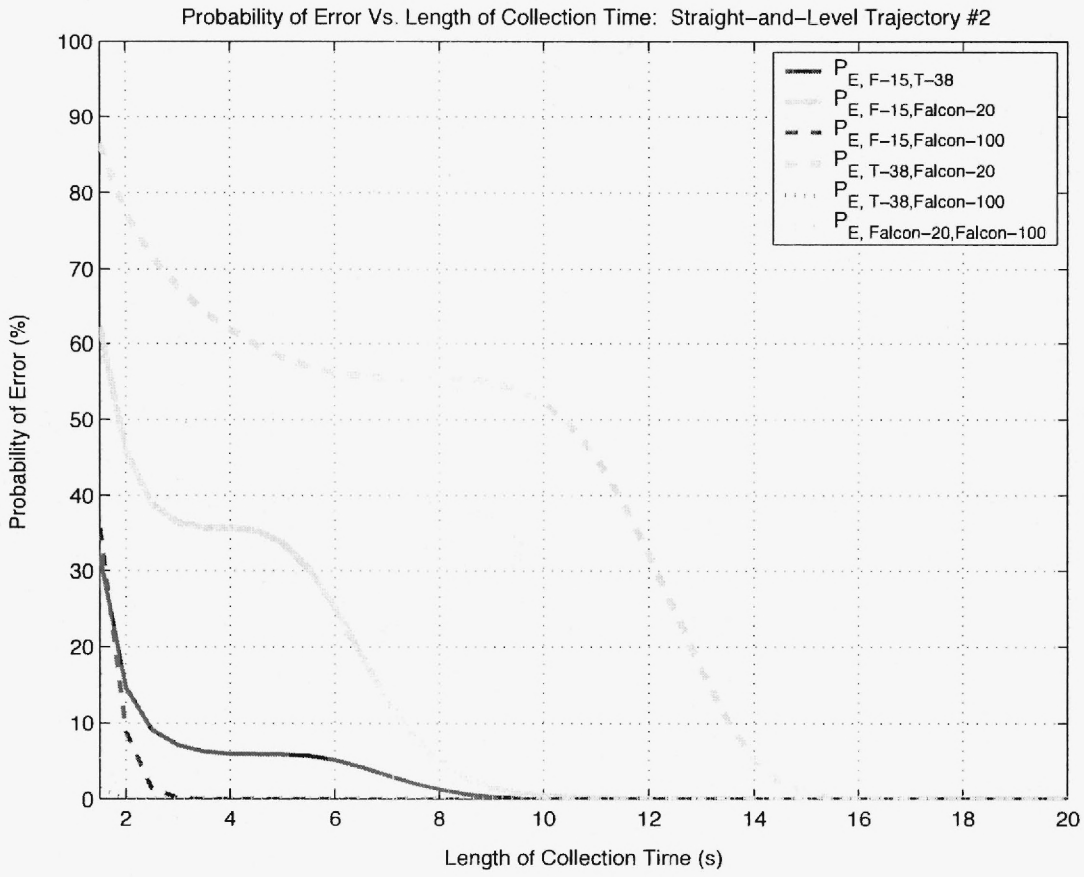


Fig. 7. Probability of Error Vs. Time: Straight-and-Level Trajectory #2, Noise Figure = 45 dB

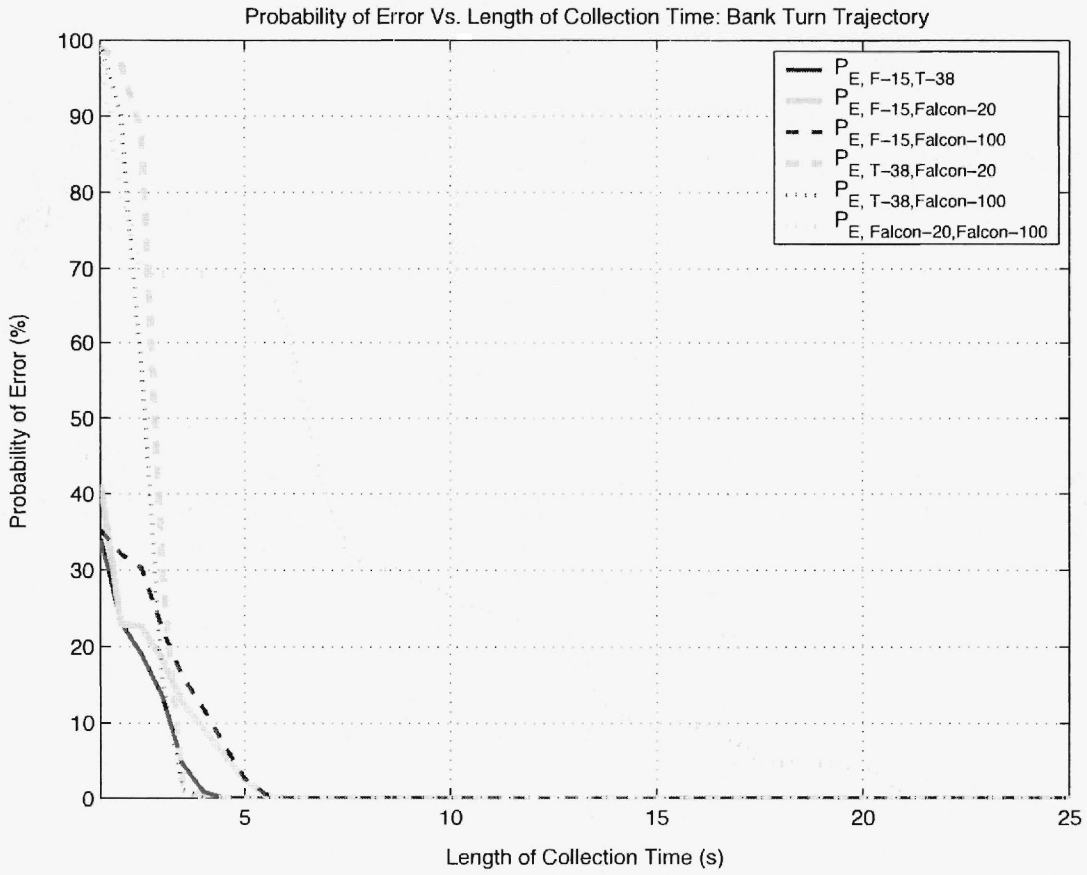


Fig. 8. Probability of Error Vs. Time: Banked Turn Trajectory, Noise Figure = 40 dB

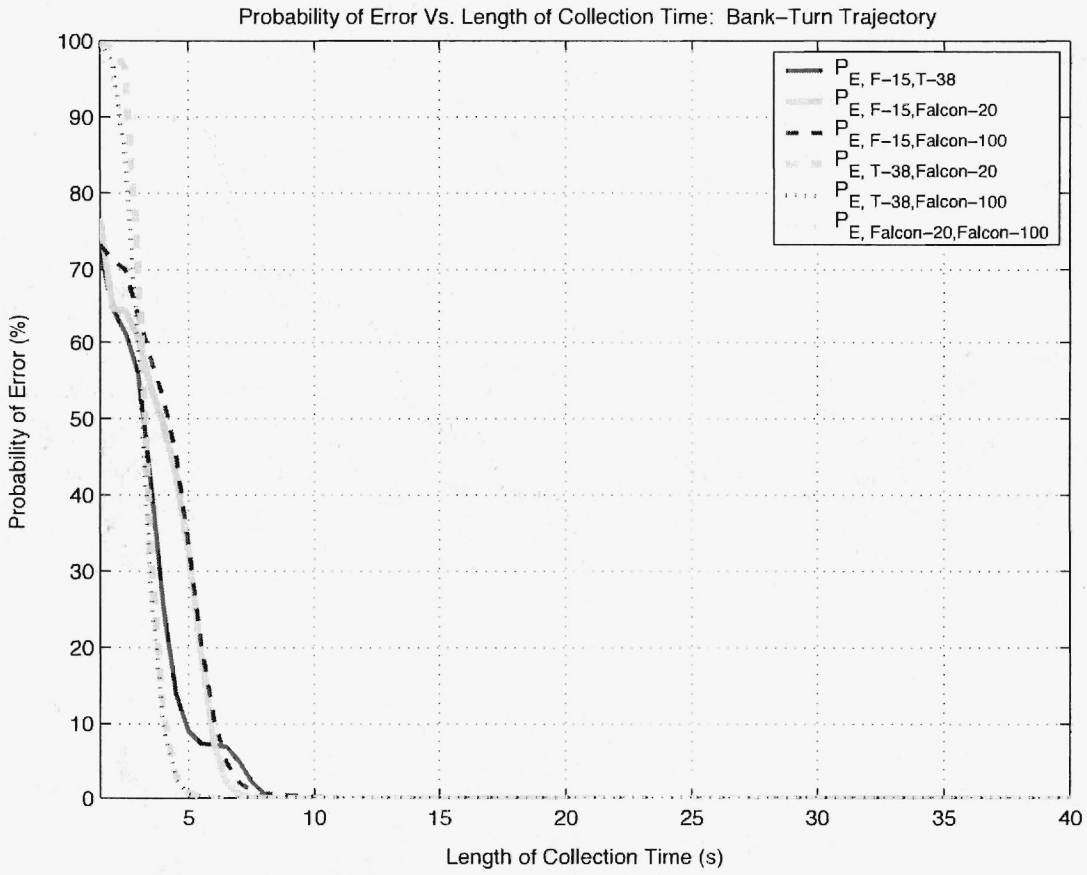


Fig. 9. Probability of Error Vs. Time: Banked Turn Trajectory, Noise Figure = 45 dB

Appendix E

L.M. Ehrman and A.D. Lanterman, "A Laplace Approximation of the Kullback-Leibler Distance Between Ricean Distributions," *IEEE Trans. on Information Theory*, to be submitted.

A Laplace Approximation of the Kullback-Leibler Distance between Ricean Distributions

Lisa M. Ehrman, *Member, IEEE* **and A.D. Lanterman,** *Member, IEEE*

Abstract— **An approximation of the Kullback-Leibler distance between Ricean distributions is derived using Laplace's method. It is found to be more accurate than a simple Gaussian approximation.**

Keywords— **relative entropy, Stein's lemma**

I. Introduction

The Rician density may be expressed as

$$p(x) = \frac{x}{\sigma^2} \exp\left(-\frac{x^2+s^2}{2\sigma^2}\right) I_0\left(\frac{xs}{\sigma^2}\right), \qquad (1)$$

The Ricean density, which reduces to a Rayleigh density if $b = 0$, arises in a number of engineering problems, from fading multipath channels in communications to modeling the radar cross section of aircraft observed with low-frequency radar [1], [2], [3].

The Kullback-Leibler distance, also called the relative entropy, is a natural information-theoretic discrepancy measure between two distributions. For instance, Stein's lemma [5] uses relative entropy to describes asymptotic behavior in detection problems.

Computing the relative entropy between two Rician distributions involves some intractible integrals. This correspondence presents closed-form approximations to the relative entropy between Ricean distributions [3], [7].

A. Derivation of the Relative Entropy Between Two Rician Densities

We will consider two Rician densities, $p(x)$ and $q(x)$, that have the same σ^2 parameters, but differing s parameters, which will be denoted as s_p and s_q. The relative entropy between two densities is given by

$$D\left(p(x)||q(x)\right) = \int_0^\infty p(x) \ln\left(\frac{p(x)}{q(x)}\right) dx, \qquad (2)$$

Substituting (1) into (2) reveals that the relative entropy between two Rician densities with the same σ^2 parameter is

$$\begin{aligned} D(p\|q) &= \left(\frac{s_q^2 - s_p^2}{2\sigma^2}\right) + \int_0^\infty p(x) \ln\left[I_0\left(\frac{xs_p}{\sigma^2}\right)\right] dx \\ &\quad - \int_0^\infty p(x) \ln\left[I_0\left(\frac{xs_q}{\sigma^2}\right)\right] dx. \end{aligned} \qquad (3)$$

The approximations presented in Sections I-B and I-C are suggested as a means of evaluating the integrals in (3).

B. A Gaussian Approximation for the Relative Entropy Between Rician Densities

The relative entropy between two Gaussian distributions, $p(x)$ and $q(x)$, is given by [4]

$$D\left(p(x)||q(x)\right) = \ln\left(\frac{v_p}{v_q}\right) + \frac{1}{2v_q^2}\left[v_p^2 + (\mu_p - \mu_q)^2\right] - \frac{1}{2}, \qquad (4)$$

where $p(x)$ is a Gaussian density with mean μ_p and variance v_p, and $q(x)$, μ_q, and v_q are defined similarly.

The mean of a Rician random variable X is given by

$$\begin{aligned} E[X] &= \sqrt{\tfrac{\pi\sigma^2}{2}} \exp\left(-\tfrac{s^2}{4\sigma^2}\right) \times \\ &\quad \left[\left(1 + \tfrac{s^2}{2\sigma^2}\right) I_0\left(\tfrac{s^2}{4\sigma^2}\right) + \left(\tfrac{s^2}{2\sigma^2}\right) I_1\left(\tfrac{s^2}{4\sigma^2}\right)\right], \end{aligned} \qquad (5)$$

and its variance is

$$\mathrm{Var}[X] = s^2 + 2\sigma^2 - \mathrm{E}^2[X]. \qquad (6)$$

If the Gaussian means and variances in (4) are set to match the Rician means and variances given in (5) and (6), then (4) approximates the relative entropy between two Rician distributions.

C. A Laplace Approximation for the Relative Entropy Between Two Rician Densities

Our proposed closed-form approximation uses Laplace's method [6] to evaluate the integrals in (3). Begin with the second integral. To apply Laplace's method, we first write

$$\int_0^\infty \frac{x}{\sigma^2} e^{\frac{-(x^2+s_p^2)}{2\sigma^2}} I_0\left(\frac{xs_p}{\sigma^2}\right) \ln\left[I_0\left(\frac{xs_q}{\sigma^2}\right)\right] dx, \qquad (7)$$

as

$$\int_0^\infty \exp[h_{pq}(x)] dx, \qquad (8)$$

where h_{pq} is defined as

$$\ln\left(\frac{x}{\sigma^2}\right) - \frac{x^2+s_p^2}{2\sigma^2} + \ln\left[I_0\left(\frac{xs_p}{\sigma^2}\right)\right] + \ln\left\{\ln\left[I_0\left(\frac{xs_q}{\sigma^2}\right)\right]\right\}. \qquad (9)$$

This work was supported by the NATO Consultation, Command, and Control Agency (NC3A), the U.S. Air Force Office of Scientific Research (grant F49620-03-1-0340), and start-up funds from the School of Electrical and Computer Engineering at the Georgia Institute of Technology.

L.M. Ehrman is with the Georgia Tech Research Institute (e-mail: lisa.ehrman@gtri.gatech.edu).

A.D. Lanterman is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Mail Code 0250, Atlanta, GA 30332 (e-mail: lanterma@ece.gatech.edu).

Taking the Taylor Series expansion of $h_{pq}(x)$ around the value of x that maximizes $h_{pq}(x)$, $\hat{x}$, results in

$$h_{pq}(x) \approx h_{pq}(\hat{x}) + \frac{1}{2}h''_{pq}(\hat{x})(x-\hat{x})^2. \quad (10)$$

Thus, (8) is approximated by

$$\exp[h_{pq}(\hat{x})]\int_0^\infty \exp\left[\frac{h''_{pq}(\hat{x})(x-\hat{x})^2}{2}\right]dx. \quad (11)$$

Taking an additional approximation of changing the lower limit in the integral from 0 to $infty$, (11) may be rearranged into a form containing the integral of a Gaussian density over its full range, which lets us approximate (11 as $\exp[h_{pq}(\hat{x})]\sqrt{-2\pi/h''_{pq}(\hat{x})}$. This provides a closed-form approximation of the third integral in (3). Similarly, the second integral in (3) may be approximated as $\exp[h_{pp}(\hat{x})]\sqrt{-2\pi/h''_{pp}(\hat{x})}$, where the definition of h_{pp} follow naturally from the definition of h_{pq}; simply substitute s_p for s_q in the last term of (9).

Our Laplace approximation for the relative entropy between Rician distributions is then given by $D(p(x)|||q(x)) =$

$$\frac{s_q^2 - s_p^2}{2\sigma^2} + \exp[h_{pp}(\hat{x})]\sqrt{\frac{-2\pi}{h''_{pp}(\hat{x})}} - \exp[h_{pq}(\hat{x})]\sqrt{\frac{-2\pi}{h''_{pq}(\hat{x})}}. \quad (12)$$

All that remains to finish this closed-form approximation is to find expressions for $\hat{x}$ and $h''_{pq}(\hat{x})$. The first derivative of $h_{pq}(x)$ is

$$h'_{pq}(x) = \frac{1}{x} - \frac{x}{\sigma^2} + \left(\frac{s_p}{\sigma^2}\right)\left[\frac{I_1\left(\frac{xs_p}{\sigma^2}\right)}{I_0\left(\frac{xs_p}{\sigma^2}\right)}\right] + \left(\frac{s_q}{\sigma^2}\right)\left\{\frac{I_1\left(\frac{xs_q}{\sigma^2}\right)}{I_0\left(\frac{xs_q}{\sigma^2}\right)\ln\left[I_0\left(\frac{xs_q}{\sigma^2}\right)\right]}\right\}. \quad (13)$$

Setting this equal to zero and using the approximation $I_p(z) \approx z$ results in

$$\frac{2}{x} - \frac{x}{\sigma^2} + \frac{s_p}{\sigma^2} = 0, \quad (14)$$

which yields a feasible solution

$$\hat{x} \approx \frac{1}{2}\left(s_p + \sqrt{s_p^2 + 8\sigma^2}\right). \quad (15)$$

Taking the derivative of (13) produces the unweildy but straightforwardly implemented expression

$$\begin{aligned} h''_{pq}(x) = {} & \frac{-1}{x^2} - \frac{1}{\sigma^2} + \left(\frac{s_p}{\sigma^2}\right)^2\left(\frac{I_1^2\left(\frac{xs_p}{\sigma^2}\right)}{I_0^2\left(\frac{xs_p}{\sigma^2}\right)}\right) \\ & + \left(\frac{s_p}{\sigma^2}\right)^2\left(\frac{I_0\left(\frac{xs_p}{\sigma^2}\right) + I_2\left(\frac{xs_p}{\sigma^2}\right)}{I_0\left(\frac{xs_p}{\sigma^2}\right)}\right) \\ & - \left(\frac{s_q}{\sigma^2}\right)^2\left(\frac{I_1^2\left(\frac{xs_q}{\sigma^2}\right)}{I_0^2\left(\frac{xs_q}{\sigma^2}\right)\ln^2\left[I_0\left(\frac{xs_q}{\sigma^2}\right)\right]}\right) \\ & - \left(\frac{s_q}{\sigma^2}\right)^2\left(\frac{I_1^2\left(\frac{xs_q}{\sigma^2}\right)}{I_0^2\left(\frac{xs_q}{\sigma^2}\right)\ln\left[I_0\left(\frac{xs_q}{\sigma^2}\right)\right]}\right) \\ & + \left(\frac{s_q}{\sigma^2}\right)^2\left(\frac{I_0\left(\frac{xs_q}{\sigma^2}\right) + I_2\left(\frac{xs_q}{\sigma^2}\right)}{I_0^2\left(\frac{xs_q}{\sigma^2}\right)\ln\left[I_0\left(\frac{xs_q}{\sigma^2}\right)\right]}\right). \end{aligned} \quad (16)$$

for the second derivative. In summary, the closed-form approximation of the relative entropy is obtained by substituting (15) and (16) into (12).

II. Comparisons

To compare the Gaussian approximation with the Laplace approximation, the s_q parameter is swept over a range of values while the s_p and σ^2 parameters are held constant. Figure 1 shows the resulting relative entropy and approximate probability of a Type II error, $\beta_{p||q} \approx \exp[-D(p||q)]$, as suggested by Stein's lemma, when $s_p = 10$, $\sigma^2 = 4$, and s_q is swept from 0 to 20. The approximation derived using the Laplace method produces results that are nearly identical to the numerically approximated results. The Gaussian approximation is not as accurate as the Laplace approximation for the smaller values of s_q. The difference becomes even more apparent if we reduce s_p to 5 and sweep s_q from 0 to 10, as shown in Figure 2.

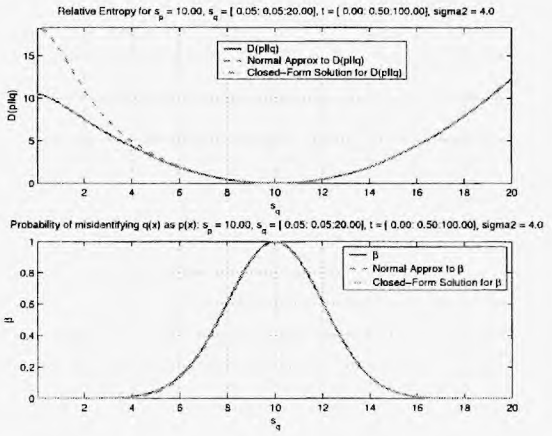


Fig. 1. $s_p = 10$, $\sigma^2 = 4$, and s_q sweeps from 0 to 20: top: $D(p||q)$, bottom: $\beta_{p||q}$

References

[1] L.M. Ehrman and A.D. Lanterman,"Automated Target Recognition using Passive Radar and Coordinated Flight Models," in *Automatic Target Recognition XIII*, SPIE Proc. 5094, Ed: F.A. Sadjadi, Orlando, FL. April 2003, pp. 196-207.

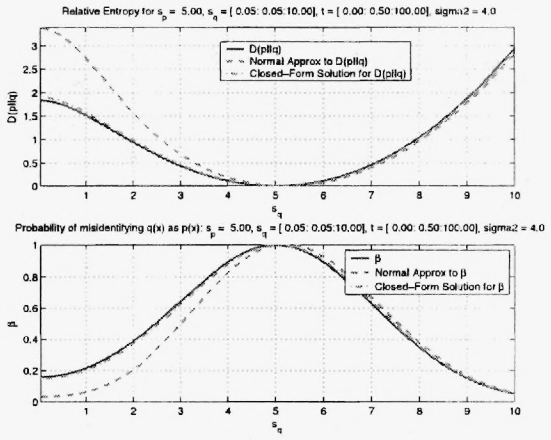


Fig. 2. $s_p = 5$, $\sigma^2 = 4$, and s_q sweeps from 0 to 10: top: $D(p||q)$, bottom: $\beta_{p||q}$

[2] L.M. Ehrman and A.D. Lanterman, "A Robust Algorithm for Automated Target Recognition using Passive Radar," in *Proc. IEEE Southeastern Symposium on System Theory*, Atlanta, GA, March 2004, pp. 102-106.

[3] L.M. Ehrman, *An Algorithm for Automatic Target Recognition Using Passive Radar and an EKF for Estimating Aircraft Orientation.* Atlanta, GA, PhD Dissertation, School of Electrical and Computer Engineering, Georgia Institute of Technology, Fall 2005.

[4] S. Kullback, *Information Theory and Statistics.* John Wiley & Sons, 1959.

[5] T.M. Cover and J.A. Thomas, *Elements of Information Theory.* John Wiley & Sons, 1991.

[6] N. DeBruijn, *Asymptotic Methods in Analysis.* Dover Publications, Inc., 1981.

[7] L.M. Ehrman and A.D. Lanterman, "Assessing the Performance of a Covert Automatic Target Recognition Algorithm," in *Automatic Target Recognition XV*, SPIE Proc. 5807, Ed: F.A. Sadjadi, Orlando, FL, April 2005, pp. 77-78.

Appendix F

A.D. Lanterman, "Continuous-Time Jump Processes, with Application to Shapes on the Lattice," *Statistics and Computing*, letter requesting major revision received Sept. 2005; manuscript undergoing revision.

Inference via Continuous-Time Jump Processes, with Application to Shapes on the Lattice

A.D. Lanterman (lanterma@ece.gatech.edu) *
School of Electrical and Computer Engineering, Georgia Institute of Technology, Mail Code 0250, Atlanta, GA 30332

Abstract. We are motivated by the need to supplement rigid target models in a pattern-theoretic automatic target algorithm with flexible models to account for clutter objects not present in the target library. A class of random sampling algorithms based on continuous-time jump processes is proposed and applied to sampling from the space of simply connected objects on the image lattice. The semigroup theory of random processes lets us show that limiting cases of certain random processes acting on discretized spaces converge to diffusion processes as the discretization is refined.

Keywords: Monte Carlo, image segmentation, pattern theory, automatic target recognition, clutter

1. Introduction

The investigations described in this paper grew out of needs that became apparent during our earlier attempts to apply Grenander's pattern theory (Grenander, 1994; Grenander, 1996) to the problem of Automatic Target Recognition (ATR) in infrared scenes (Lanterman et al., 1997a; Lanterman, 1998). In our formulation, targets were characterized by three-dimensional Computer-Aided Design (CAD) models along with mathematical representations of the varying intensities of the facets (Lanterman, 2000). A Monte Carlo algorithm based on jump-diffusion dynamics (Miller et al., 1997), where the jumps added and removed individual targets and the diffusions refined their positions and orientations, was used to sample from a Bayesian posterior incorporating the degradations of the sensor.

Our initial work assumed that the background was uniform, and that the scene consistent only of known targets. Of course, the most challenging aspect of ATR, and computer vision in general, is clutter. When presented with a large building, our simple jump-diffusion algorithm would "birth" tanks into the building, and then pack them as tightly as possible to cover the building structure. Since we only told the algorithm about tanks, it felt compelled to explain *everything*

* This research was supported by ARO DAAH04-95-0494, ARO/AASERT DAAH04-94-G-0209, and AFOSR F49620-03-1-0340.

in the scene using tanks! Many traditional ATR systems in general use outlier rejection techniques to try to screen out clutter. Hoping to stick more closely to Grenander's framework, we decided to add flexible models to the parameter space, thus giving the algorithm the option of describing targets using those flexible models (Lanterman et al., 1997b). Our ultimate goal is an algorithm that can readily move back and forth between structured representations, such as specific CAD models, and unstructured representations.

The literature offers rich menagerie of flexible models, such as varieties of snakes (Kass et al., 1988) and balloons (Kichenessamy et al., 1997; Figueiredo et al., 1997), and deformable templates ranging from hands (Grenander and Keenan, 1993; Grenander et al., 1990) to potatoes (Grenander and Manbeck, 1993). The insights of Zhu and Yuille (1996) have harmonized many of these approaches with one another and with more earthy pixel-based image segmentation approaches.

Translating the mathematical beauty of different shape models to the cold, hard, unforgiving world of the computer is fraught with practical discretization difficulties and headaches associated with maintaining and updating list-like data structures. Ingenious level-set methods (Sethian, 1996) have helped ease some of the pain, but even these elegant techniques sometimes require the clever coders to have a few tricks up their computational sleeve.[1] Many stalwart researchers have conquered these assorted challenges.

For our ATR work, we plead for a simpler formulation of flexible shapes that would not require the calisthenics described in the previous paragraph. Our plea is answered by simply connected shapes on the pixel lattice, which we viscerally call *blobs*. To sample from the space of blobs, we develop a Monte Carlo algorithm based on continuous-time jump processes. This was chosen to make the blob inference algorithm readily compatible with the larger, more complex jump-diffusion process for ATR in infrared scenes we are currently working to embed it in. The first goal of this paper is to present this algorithm for random sampling from the space of blobs.

After implementing the algorithm for blobs, it became apparent that continuous-time jump processes might be of interest in other Monte Carlo applications as well. Hence, the second goal of this paper is to relate some specific continuous-time jump processes to diffusion processes, thereby linking two of main tools in the Monte Carlo designer's bag of tricks.

[1] The author would like to thank Anthony Yezzi of the School of Electrical and Computer Engineering at the Georgia Institute of Technology for letting him sit in on Prof. Yezzi's special topics class on Partial Differential Equation Methods in Image Processing. This class helped clear up numerous areas of confusion.

1.1. A Simple Example of the Challenge of Clutter

1.1.1. *Shape Wars*

To illustrate the need for clutter modeling in a pattern-theoretic framework in the Grenander style, consider a simplified example of detecting a tank, shown in the left panel of Figure 1, of known pose and radiant intensity against a background of known intensity. In this example, a Poisson noise model will be used. The objects have a Poisson intensity of 40 and the background a Poisson intensity of 20. Suppose the algorithm may also encounter a boulder, shown in the middle panel of Figure 1, but the system is not expecting to see anything except the tank. To detect the tank, we can compute the loglikelihood of the tank and compare it against the loglikelihood of only background. If we do this for the boulder data shown in the right panel of Figure 1, we find the loglikelihood of there not being a tank is 175204, and the likelihood of there being a tank a 176112. Hence, the algorithm insists on calling the boulder a tank.

Figure 1. Left and middle panels display templates of a tank and a boulder. Right panel shows noisy image of the boulder.

This motivates extending the representation beyond the targets of immediate interest with flexible models that can accommodate other objects that might be encountered, such as this boulder. As described in the introduction, this paper explores *blobs*, defined as simply-connected objects on the image lattice, with connectivity defined along the four compass directions. (Of course, other flexible models could be used as well. The vital point here is that any ATR system must incorporate some kind of generic shape model if it is to be robust against clutter, not that blobs are the only viable approach.)

A high probability blob matching the data is shown in Figure 2. The loglikelihood of the tank is 176112, but the loglikelihood of the blob is 176548. In essence, the algorithm declares that something interesting is there, but that it does not know what it is. We do not wish to immediately discard such clutter objects, since their presence might be of interest, even if they do not match a target present in the ATR system's library.

Figure 2. A blob that matches the data of the boulder with high probability.

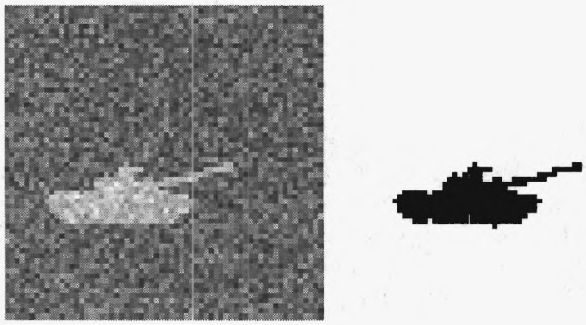


Figure 3. Left panel shows noisy image of a tank. Right panel shows a blob that matches the tank data with high probability.

1.1.2. *The Boulder Strikes Back*

This subsection briefly mentions a difficulty that we will address in future work. While the blobs effectively handle the boulder, new trouble arises when we return to the original tank. Suppose the algorithm encounters the tank data shown in the left panel of Figure 3. The loglikelihood of a tank is 181359, but the loglikelihood of the matching blob shown in the right panel of Figure 3 is 181367. The algorithm wants to call the tank a blob, even though it is really a tank. This is a manifestation of the model order estimation problem. Since the blob has more degrees of freedom than the tank, it can twist and turn to match minute details in the data. It wields an unfair advantage over the simpler tank model. An approach such as Rissanen's minimum description length framework (Barron et al., 1998) levels the playing field, as the blob model must pay a price for its flexibility; this observation motivated our investigations into MDL begun in Lanterman (2001).

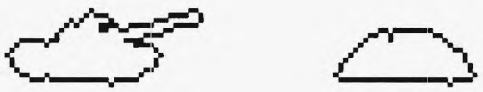


Figure 4. Boundaries of blobs that match the tank (left) and boulder (right) data with high probability.

1.1.3. *Return of the Tank*

As a simple example illustrating how the problem of the previous subsection might be tackled, one straightforward approach is to invoke a penalty based on a chain-length encoding of the boundaries, in the spirit of Leclerc (1989). Consider the boundaries shown in Figure 4. At each point on the boundary, there are five choices: one can continue forward, bend 45 degrees to the left or right, or bend 90 degrees to the left or right. A simple penalty could employ ln(5) *nats* (natural logarithmic units of information) for each point along the border.

Using such a penalty, the penalized likelihood of a blob for the tank data is 181206, while the likelihood of the tank is 181359, so the tank hypothesis justifiably wins. When we return to the boulder data, we find the likelihood of a tank is 176112, while the penalized likelihood of a blob is 176451. The description length penalty has lowered the favorability of the blob slightly, but not enough for the algorithm to start claiming that the boulder is a tank.

1.2. Two Kinds of Convergence

This paper invokes the term *convergence* in two distinct ways. The first sense of convergence is the convergence of the marginal distribution of the random processes, over time, to the posterior distribution under investigation. The "time" here refers not to the physical time over which we collect data, but an "algorithmic time" associated with the fictitious process we construct. Previous studies involving jump-diffusion algorithms and related Gibbs and Metropolis-Hastings sampling algorithms have focused on this sort of convergence.

The second sense of convergence is the convergence of a sequence of different processes to some limiting process, with the convergence

being taken over the entire process. This is a more complex issue. While the convergence described in the proceeding paragraph is over simple spaces such as $\Re^n$, or perhaps groups like $SO(n)$, the convergence of entire processes deals with abstract infinite-dimensional spaces, and defining these notions properly requires considerable care.

Our Swiss army knife for the study of convergence will be semigroup theory, particularly the *infinitesimal generators* that characterize those semigroups. To show that a chosen process converges to the posterior distribution, it suffices to show that the integral of the generator against the target distribution is zero, as formulated precisely in Section 5. This sort of argument has been a staple of research into pattern-theoretic jump-diffusion algorithms, beginning with Section 3 of the seminal monograph by Grenander and Miller (1991), through Theorem 1 of Grenander and Miller (1994) and the appendices of Miller et al., Miller et al. (1995, 1997).

To show convergence in the second sense – convergence of entire processes – we will show that their associated generators converge to the generator of the limiting process. This implies convergence of semigroups, which in turn implies convergence of processes in the *Skorohod topology*, defined in Section 5.2.1. This is one of the approaches suggested in the preface of Ethier and Kurtz (1986). They suggest two other techniques, one grounded in martingale theory, the other involving the solution of stochastic equations. The generator/semigroup approach appears to require the least mathematical machinery of the three techniques, so it is the one we adopt here. Still, much labor is needed to make this rigorous. We tackle this in Section 5.2.2.

The codex by Ethier and Kurtz (1986) may be the most extensive treatise available on the detailed characterization of Markov processes and their convergence via semigroup theory. Much of the material it contains is not readily available elsewhere, and it is quite challenging even for those familiar with random process theory, so we will spend some time reviewing it here. Some of the theorems cited are more powerful (and hence more opaque) than we need for our present purposes; hence we will state the theorems as simplified special cases whenever possible.

2. A Formal Definition Blobs

This section is intended primarily to fix notation; the definitions given here will not be a surprise. Let $\mathcal{P} \subset \mathbf{Z}^2$ denote a set of pixels on the image detector plane, such as a video camera or the infrared camera of

interest in our specific ATR application. For rigor, we define *blobs* via the following:

DEFINITION 1. *A* binary lattice pattern *on $\mathcal{P}$ is a function $f : \mathcal{P} \to \{0,1\}$.*

DEFINITION 2. *The* primary compass directions *are*

$$PCD = \{(0,1),(0,-1),(1,0),(-1,0)\}. \tag{1}$$

DEFINITION 3. *A binary lattice pattern is a* blob *if for all $p_1, p_2 \in \mathcal{P}$ such that $f(p_1) = f(p_2)$, there exists a sequence of directions $(d_1, \ldots d_N) \in PCD^N$ such that*

$$p_2 = p_1 + \sum_{i=1}^{N} d_i \tag{2}$$

and

$$f(p_2) = f(p_1 + \sum_{i=1}^{J} d_i) \text{ for all } J = 1, ..., N, \tag{3}$$

i.e., for any two foreground points, there is a path on the lattice that connects them while staying in the foreground, and for any two background points, there is a path on the lattice that connects them while staying in the background.

It will be convenient to define a predicate: $IsBlob[f] = 1$ if f is a blob, 0 otherwise. With this, define the following:

DEFINITION 4. *The set of blobs on $\mathcal{P}$ will be called $\mathcal{B}$ = blobs.*

$$\mathcal{B} = \{f : \mathcal{P} \to \{0,1\} : IsBlob[f] = 1\}. \tag{4}$$

DEFINITION 5. *The* boundary operator $b : \mathcal{B} \to \mathcal{B}$, *which maps a blob f to its* boundary $b(f)$, *is given by*

$$b(f)(p) = \bigvee_{d \in PCD} f(p) \oplus f(p+d), \tag{5}$$

where $\oplus$ represents the "exclusive-or" operation and $\vee$ represents the "or" operation.

The following similar definitions will be useful in Section 7 when we design Monte Carlo algorithms for sampling from $\mathcal{B}$.

DEFINITION 6. *The* exterior boundary $eb(f)$ *of a blob f is given by*

$$eb(f)(p) = not[f(p)] \wedge b(f)(p), \tag{6}$$

i.e. all boundary points in the foreground.

DEFINITION 7. *The* interior boundary $ib(f)$ *of a blob* f *is given by*

$$ib(f)(p) = f(p) \wedge b(f)(p), \tag{7}$$

i.e., all boundary points in the background. Here, $\wedge$ *represents the "and" operation.*

DEFINITION 8. *The* allowable additions $aa(f)$ *of a blob* f *is the subset of the exterior boundary such that adding a point anywhere along* $aa(f)$ *yields another blob:*

$$aa(f)(p) = eb(f)(p) \wedge IsBlob[f \vee \delta(p)], \tag{8}$$

where $\delta(p)$ *is the single-point blob with 1 at* p *and* 0 *elsewhere.*

DEFINITION 9. *The* allowable deletions $ad(f)$ *of a blob* f *is the subset of the interior boundary such that deleting a point anywhere along* $ad(f)$ *yields another blob:*

$$ad(f)(p) = ib(f)(p) \wedge IsBlob[f - \delta(p)]. \tag{9}$$

Adding a point in the allowable additions will not form a "hole" in the blob, and deleting a point in the allowable deletions will not disconnect the blob into two separate blobs.

A simpler, computationally convenient characterization of $aa(f)$ and $ad(f)$ is helpful. At a given point p, imagine tracing along its eight neighbors; for that point to be in the allowable region, two and only two transitions, one from 0 to 1 and one from 1 to 0, should be observed as we move along the neighbors. Reusing notation somewhat and adding 1's (true) and 0's (false) as if they were integers, define $TwoCh(f)(p)$ to be 1 if

$$\begin{aligned} 2 \;=\; & [f(p+E) \oplus f(p+SE)] + [f(p+SE) \oplus f(p+S)] \\ & + [f(p+S) \oplus f(p+SW)] + [f(p+SW) \oplus f(p+W)] \\ & + [f(p+W) \oplus f(p+NW)] + [f(p+NW) \oplus f(p+N)] \\ & + [f(p+N) \oplus f(p+NE)] + [f(p+NE) \oplus f(p+E)], \end{aligned}$$

and 0 otherwise, where in (row,column) notation, $E = (0,1)$, $SE = (-1,1)$, $S = (-1,0)$, etc., and $TwoCh$ stands for "two changes."

Then straightforward (albeit tedious) enumeration of cases shows that

$$aa(f) \;=\; eb(f) \wedge TwoCh(f), \tag{10}$$

$$ad(f) \;=\; ib(f) \wedge TwoCh(f). \tag{11}$$

3. An Example Likelihood Model

3.1. Model for the Detector

A detector with elements arranged on a Cartesian lattice is assumed. Enumerate the detector elements as $p \in \mathcal{P} \subset \mathbf{Z}^2$. In most applications $\mathcal{P}$ is a rectangular subset.

We suppose an "ideal image" $\lambda : \mathcal{P} \to [0, \infty)$, observed by charge-coupled-device camera, produces Poisson-distributed data $y : \mathcal{P} \to 0, 1, 2...$ with mean given by λ. In the infrared case, λ is proportional to the radiant intensity of objects in the scene. The associated log-likelihood of the data is (Snyder and Miller, 1991)

$$L(y|\lambda) = -\sum_k \lambda(k) + \sum_k y(k) \ln \lambda(k). \tag{12}$$

Some non-ideal phenomena such as camera point-spread, nonuniform detector response, and readout noise are not incorporated in (12). These second-order effects are described in Snyder et al. (1993).

3.2. Intensity Models for Blobs and Background

For the blobs and the background, little *a priori* knowledge of their intensities is available. Thus, we model their intensities as unknown, nonrandom parameters. This was the technique employed to deal with the nuisance parameters in the HANDS study (Grenander et al., 1990). There, the nuisance parameters were the intensities of the hand and of the background.

Taking the intensity across the shape to be uniform, a Poisson noise model implies that the Maximum-Likelihood (ML) estimate is just the mean of the pixels. The ML estimate $\hat{\lambda}_i$ of the intensity λ_i of a region R_i containing N_i pixels is simply the average of the data values $y(\cdot)$ in that region:

$$\hat{\lambda}_i = \frac{1}{N_i} \sum_{k \in R_i} y(k) = \frac{Y_i}{N_i}. \tag{13}$$

4. Background on Stochastic Processes

4.1. Algorithmic Context

Geman and Hwang (1987), among others, have proposed simulating Langevin diffusions defined by the SDE

$$dX(t) = \frac{1}{2}E'(X(t))dt + dW(t), \tag{14}$$

where $W(t)$ is a Wiener process, to sample from densities of the form $\pi(x) = \exp[E(x)]/Z$. The Langevin diffusion is essentially a gradient ascent with an additional random term. To accommodate scenes of varying dimension, Grenander and Miller (1991) combined diffusions with jump processes to move between subspaces.

Figure 5 show an example of a Langevin diffusion process refining the estimate of the orientation (Lanterman et al., 1997a) of an M60 in an image provided by Dr. Richard Sims of the U.S. Aviation and Missile Command (AMCOM). In this example, radiant intensities of target facets were inferred from the data and assumed known in the analysis. They were extracted from the data; the turret was assumed to possess one uniform intensity, and the body another, both estimated from the actual data prior to running the diffusion. The background intensity, also estimated from the data, was assumed uniform and known. The AMCOM image was collected from a infrared camera mounted aboard a helicopter. Altitude and range to target were collected along with the data, permitting rough reconstruction of the viewing geometry. The position of the target was assumed known, with the diffusion operating on the orientation parameter.

Langevin diffusions must be discretized for computer implementation, and the discretized process only approximately maintains detailed balance. Besag (1994) suggests this can be readily remedied by performing a Metropolis-Hastings acceptance/rejection step (detailed in Section 7.1) after making the discretized Langevin diffusion; the resulting discrete-time Markov chain algorithm incorporates all the advantages of the Langevin diffusion approach, namely the analogy with the gradient search (natural for continuous variables) and completely parallel site updating. Thus diffusion techniques fit nicely within the Metropolis-Hastings umbrella. Roberts and Rosenthal (1998) explore this approach in depth and deduce optimal step sizes.

This section and the remaining sections offer two primary take-home messages in this vein:

1. Metropolis-Hastings algorithms and other discrete-time Markov chain Monte Carlo algorithms have been the subject of much investigation in the statistical literature. In Section 6.2, we introduce a

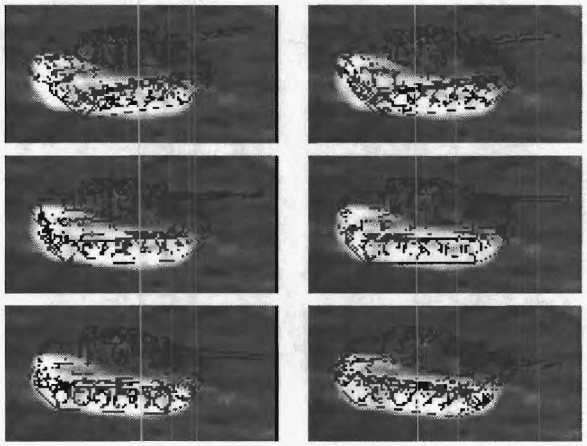


Figure 5. An example diffusion process refining the orientation of the M60 in the AMCOM data.

continuous-time analogue of a conditional Gibbs sampler, where the process jumps on exponential times, with mean proportional to the inverse of the total probability contained in a neighborhood around the current state. Compared with discrete-time MCMC techniques, such continuous-time jump processes appear to have remained largely unexplored by the random sampling community (although they are well-known as models for physical phenomena: see, for instance, Chapters 4 and 5 of Gillespie (1992).) We apply this approach to derive samplers operating on the space of blobs defined in Section 2.

2. If derivatives are difficult to compute, Metropolis-Hasting proposals can be made using a density centered around the old value, such as a Gaussian. The acceptance/rejection step ensures that we still achieve the target distribution. Most intriguingly, Gelfand and Mitter (1991) have shown that certain continuous-time interpolations of some Metropolis sampling algorithms, such as the Gaussian proposals mentioned here, weakly converge to Langevin diffusions with the same stationary distribution. Thus, even if the gradient is not explicitly employed, it implicitly guides the inference in a limiting sense. The difficulty of computing derivatives in recognizing objects in visual aerial images for the U.K. Defence Evaluation and Research Agency led Reno (1998) to construct jump-diffusion algorithms using such an approach. We explore variations of this idea in Section 8, where algorithms that operate on discretized

sample spaces are shown to converge to different kinds of diffusions as the discretization is refined.

4.2. Operator Semigroups and their Infinitesimal Generators

We begin with some definitions relating to operator semigroups.

DEFINITION 10. *(p. 6,* Ethier and Kurtz *(1986)) A one-parameter family* $\{T(t) : t \geq 0\}$ *of bounded linear operators on a Banach space* L *is called a* semigroup *if* $T(0) = I$ *and* $T(s+t) = T(s)T(t)$ *for all* $s, t \geq 0$. *A semigroup* $T(\cdot)$ *is* strongly continuous *if* $\lim_{t \to 0} T(t)f = f$ *for every* $f \in L$. *It is a* contraction *semigroup if* $\|T(t)\| \leq 1$ *for all* $t \geq 0$.

DEFINITION 11. *(p. 8,* Ethier and Kurtz *(1986)) The infinitesimal generator of a semigroup* $T(\cdot)$ *on* L *is the linear operator* A *defined by*

$$Af = \lim_{t \to 0} \frac{T(t)f - f}{t}. \tag{15}$$

The domain $\mathcal{D}(A)$ *of* A *is the subspace of all* $f \in L$ *for which this limit exists.*

A semigroup and its generator can be related via a differential equation according to the following theorem:

THEOREM 1. *(Prop. 1.5(b), p. 9,* Ethier and Kurtz *(1986)) Let* $T(\cdot)$ *be a strongly continuous semigroup on* L *with generator* A. *If* $f \in \mathcal{D}(A)$ *and* $t \geq 0$, *then* $T(t)f \in \mathcal{D}(A)$ *and*

$$\frac{d}{dt} T(t)f = AT(t)f. \tag{16}$$

By Proposition 2.9 on p. 15 of Ethier and Kurtz (1986), if two strongly continuous contraction semigroups $T(\cdot)$ and $S(\cdot)$ have the same generator, then they are the same for all $t \geq 0$. Hence, the generator offers a unique characterization.

In the sequel, it will be helpful to have the concept of the *core* of an operator.

DEFINITION 12. *(p. 16,* Ethier and Kurtz *(1986)) A linear operator* A *on* L *is said to be* closable *if it has a closed linear extension. If* A *is closable, then the* closure $\bar{A}$ *of* A *is the minimal closed linear extension of* A; *more specifically, it is the closed linear operator* B *whose graph is the closure (in* $L \times L$*) of the graph of* A.

DEFINITION 13. *(p. 17,* Ethier and Kurtz *(1986)) Let A be a closed linear operator on L. A subspace D of $\mathcal{D}(A)$ is said to be a* core *for A if the closure of the restriction of A to D is equal to A.*

4.3. The Operator Semigroup of a Markov Process

For a Markov process with transition measure $P_t(x, dy)$, define the corresponding semigroup to be

$$T(t)f(x) \stackrel{\text{df}}{=} \int f(y) P_t(x, dy). \tag{17}$$

5. The Two Kinds of Convergence

5.1. Convergence to a Stationary Distribution

Stochastic processes can converge in a wide variety of ways. The kind of convergence of interest here is called weak convergence.

DEFINITION 14. *(p. 155,* Ethier and Kurtz *(1986)) $\bar{C}(E)$ is the space of bounded continuous functions on a metric space E.*

DEFINITION 15. *(Eqn. 3.2, p. 108 of* Ethier and Kurtz *(1986) or p. 271 of* Durrett *(1996)) X_n converges weakly to X, denoted $X_n \Rightarrow X$, if $E[f(X_n)] \to E[f(X)]$ for all $f \in \bar{C}(S)$.*

We are interested in formulating Markov processes that have a desired *stationary distribution*. In our pattern-theoretic application, this is the Bayesian posterior distribution. The definitions given by Ethier and Kurtz are characterized in terms of solutions to the *martingale problem* (Ethier and Kurtz (1986), p. 173), which is a highly technical way of characterizing Markov processes. For our present purposes, it will suffice to say that a process X is a *solution to the martingale problem* for (A, μ) if μ is the distribution of $X(0)$ and the transition probability P_t of X is specified by a semigroup T corresponding to the generator A. A precise technical definition is given on pp. 173-174 of Ethier and Kurtz (1986). If any two solutions X and Y of a martingale problem have the same finite-dimensional distributions, the problem is said to be *well-posed* (Ethier and Kurtz (1986), p. 182). The existence of such solutions X for the classes of jump and diffusion processes discussed in this paper, as well as their well-posedness, are addressed in Chapters 5 and 6 of Srivastava (1996) (see, for instance, Theorem 6.3).

DEFINITION 16. *(*Ethier and Kurtz *(1986), p. 238) Let $A \subset B(E) \times B(E)$, where $B(E)$ is the set of bounded measurable real-valued functions on E, and suppose the martingale problem for A is well-posed. Then $\mu \in \mathcal{P}(E)$ is a* stationary distribution *(also called an* invariant distribution*) for A if every solution X of the martingale problem for (A, μ) is a stationary process, i.e.*

$$Pr\{X(t+s_1) \in \Gamma_1, X(t+s_2) \in \Gamma_2, \ldots, X(t+s_k) \in \Gamma_k\} \qquad (18)$$

is independent of $t \geq 0$ for all $k \geq 1$, $0 \geq s_1 < \cdots < s_k$, and $\Gamma_1, \ldots, \Gamma_l \in \mathcal{B}(E)$, where $\mathcal{B}(E)$ are the Borel sets of E.

As foreshadowed in the introduction, we can show the existence of a stationary distribution for a Markov process via a necessary condition relating to its generator.

THEOREM 2. *(Prop. 9.2, p. 239,* Ethier and Kurtz *(1986)) Suppose A generates a strongly continuous contraction semigroup $T(\cdot)$ on a closed subspace $L \subset B(E)$, L is separating, and the martingale problem for A is well-posed. If D is a core for A and $\mu \in \mathcal{P}(E)$, then μ is a stationary distribution for A if and only if $\int Af d\mu = 0$ for all $f \in D$.*

5.2. Convergence of Sequences of Processes

5.2.1. *The Skorohod Topology on $D_E[0, \infty)$*

We begin with specifying $D_E[0, \infty)$, the space of sample paths of the Markov processes of interest.

DEFINITION 17. *(p. 116 of* Ethier and Kurtz *(1986) or p. 293 of* Durrett *(1996)) $D_E[0, \infty)$ is the space of functions from $[0, \infty)$ into E that are right continuous (for each $t \geq 0$, $\lim_{s \to t^+} x(s) = x(t)$) and have left limits (i.e. the $\lim_{s \to t^-} x(s) \stackrel{\text{df}}{=} x(t-)$ exists; by convention, $x(0-) = x(0)$.)*

To answer questions of convergence, a topology on $D_E[0, \infty)$ is needed. Of particular interest is the Skorohod topology, based on the distance defined on p. 117 of Ethier and Kurtz (1986). Here we present the equivalent, but somewhat simpler, definition from p. 294-295 of Durrett (1996).

DEFINITION 18. *Let Λ be the collection of strictly increasing continuous mappings of $[0, \infty)$ onto itself. (Note $\lambda(0) = 0$ and $\lim_{t \to \infty} \lambda(t) =$*

∞.) *The Skorohod distance is given by*

$$d(x,y) = \inf\{\epsilon > 0 : \exists \lambda \in \Lambda \text{ s.t. } \sup_{t>s\geq 0} \left|\frac{\lambda(s)-\lambda(t)}{s-t}\right| \leq \epsilon \text{ and } \sup_{t\geq 0} |x(t) - y(\lambda(t))| \leq \epsilon\}. \quad (19)$$

Arguments on p. 117-118 of Ethier and Kurtz *(1986) show that $d(x,y)$ is indeed a metric and forms a complete metric space. The topology induced by d is called the* Skorohod topology.

The expression (19) is rather unintuitive. The following theorem asserts that convergence in the Skorohod topology implies convergence of finite distributions, offering a more concrete interpretation.

THEOREM 3. *(Thm. 7.8, p. 131,* Ethier and Kurtz *(1986)) Let E be separable and let X_n, $n = 1, 2, \ldots$, and X be processes with sample paths in $D_E[0,\infty)$. If $X_n \Rightarrow X$ in the Skorohod topology (remember that n here indexes entire processes, not individual time points of a single process), then*

$$(X_n(t_1), \ldots, X_n(t_k)) \Rightarrow (X(t_1), \ldots, X(t_k)) \quad (20)$$

for every finite set $\{t_1, \ldots, t_k\} \subset D(X)$, where $D(X) \stackrel{\text{df}}{=} \{t \geq 0 : Pr\{X(t) = X(t-)\} = 1\}$.

5.2.2. *Convergence of Processes via Convergence of Generators*
First, note that convergence of generators implies convergence of their associated semigroups.

THEOREM 4. *(Thm. 6.1, p. 28,* Ethier and Kurtz *(1986)) For $n = 1, 2, \ldots$, let $T_n(t)$ and $T(t)$ be strongly continuous contraction semigroups on L_n and L with generators A_n and A. Let D be a core for A. Then the following are equivalent:*
a) For each $f \in L$, $T_n(t)f \to T(t)f$ for all $t \geq 0$.
b) For each $f \in D$, there exists $f_n \in \mathcal{D}(A_n)$ for each $n \geq 1$ such that $f_n \to f$ and $A_n f_n \to A$.

Next, note that convergence of semigroups implies convergence of their associated Markov processes. We will need a few definitions.

DEFINITION 19. *(p. 164-165,* Ethier and Kurtz *(1986)) $\hat{C}(E)$ is the space of continuous functions vanishing at infinity with norm $\|f\| \stackrel{\text{df}}{=} \sup_{x\in E} |f(x)|$.*

DEFINITION 20. *(p. 165,* Ethier and Kurtz *(1986)) A semigroup $T(\cdot)$ on $\hat{C}(E)$ is* positive *if $T(t)$ maps nonnegative functions to nonnegative functions for each $t \geq 0$.*

DEFINITION 21. *(p. 166,* Ethier and Kurtz *(1986)) A strongly continuous, positive, contraction semigroup on $\hat{C}(E)$ whose generator is conservative is called a* Feller semigroup.

In Definition 21, the term *conservative* is used. This is a technical condition defined on p. 166 of Ethier and Kurtz (1986). Here, it will be sufficient to follow Srivastava's approach (p. 64, Srivastava (1996)) and note that (as stated on p. 166 of Ethier and Kurtz (1986)) for semigroups specified by transition functions, conservation is ensured by the fact that $P_t(x, E) = 1$ (since E is the entire space the process can take values in at each time instant).

Now we can proceed with the theorem.

THEOREM 5. *(Thm. 2.5, p. 167,* Ethier and Kurtz *(1986)) Let E be locally compact and separable. For $n = 1, 2, \ldots$ let $T_n(\cdot)$ be a Feller semigroup on $\hat{C}(E)$, and suppose X_n is a Markov process corresponding to $T_n(\cdot)$ with sample paths in $D_E[0, \infty)$. Suppose that $T(\cdot)$ is a Feller semigroup on $\hat{C}(E)$ and that for each $f \in \hat{C}(E)$,*

$$\lim_{n \to \infty} T_n(t) f = T(t) f, t \geq 0. \tag{21}$$

If $X_n(0)$ has limiting distribution $\nu \in \mathcal{P}(E)$, then there is a Markov process X corresponding to $T(\cdot)$ with initial distribution ν and sample paths in $D_E[0, \infty)$, and $X_n \Rightarrow X$ in the Skorohod topology.

6. Continuous-Time Jump Processes

6.1. Definitions

A continuous-time Markov jump process is characterized by a jump intensity measure $q(x, dy)$, which is given according to

$$q(x, Y) = \lim_{t \to 0} \frac{1}{t} [P_t(x, Y) - I_Y(x)], \tag{22}$$

where $I_Y(x)$ is an indicator function that is 1 when $x \in Y$ and 0 otherwise.

The generator for a jump process is

$$Af(x) = -q(x) f(x) + q(x) \int Q(x, dy) f(y), \tag{23}$$

where

$$q(x) = \int_{\mathcal{X}} q(x, dy) \tag{24}$$

and

$$Q(x, dy) = \frac{q(x, dy)}{q(x)}. \tag{25}$$

As described on p. 163 of Ethier and Kurtz (1986), a continuous-time jump process may be simulated by generating a discrete-time Markov chain, in which we wait a certain random amount of time at each stage in the chain as follows. Beginning from a state x_0 at time $\tau = t_0$, a sample path of the process $X(\tau)$ is constructed as follows:

1. Draw an exponentially distributed random variable w_i variable with mean $1/q(x_i)$.
2. Let $t_{i+1} = t_i + w_i$. $X(\tau) = x_i$ for $\tau \in [t_i, t_{i+1})$.
3. Draw $y = x_{i+1}$ from the *transition distribution* $Q(x, dy)$.
4. Assign $i \leftarrow i + 1$ and goto step 1.

6.2. A Conditional-Gibbs Sampler Subordinated to a Markov Process

We now consider a random-sampling algorithm based on a continuous-time jump process analogous to Corollary 1 of Theorems 1 and 2 of Grenander and Miller (1994). In Grenander and Miller (1994), the process diffuses between jumps. In this section, we will keep the state constant between jumps.

Suppose we restrict set of potential transitions to a neighborhood around the current state x, denoted $\mathcal{N}(x)$. We require the neighborhood to have the following properties:

1. Reversibility: $y \in \mathcal{N}(x)$ if and only if $x \in \mathcal{N}(y)$. Written in terms of indicator functions, we have $I_{\mathcal{N}(x)}(y) = I_{\mathcal{N}(y)}(x)$.
2. Connectedness: For any x and y, there exists a finite sequence of states $z_1, \ldots, z_m$ such that $z_1 \in \mathcal{N}(x), \ldots, y \in \mathcal{N}(z_m)$.

Suppose we define our jump transition intensity according to the posterior density, restricted to the neighborhood:

$$q(x, dy) = \pi(y) I_{\mathcal{N}(x)}(y) dy \tag{26}$$

and

$$q(x) = \int_{\mathcal{N}(x)} \pi(y) dy. \tag{27}$$

Theorem 2 may be used to show that $\pi(x)dx$ is an invariant measure of the resulting jump process. Furthermore, the connectedness of the neighborhoods implies the irreducibility of the underlying chain, which in turn implies that the stationary measure is unique and that the process (starting from any initial distribution) converges in total variation norm to the invariant measure (see p. 14 of Grenander and Miller (1991) or p. 73-74 of Srivastava (1996) for details and further discussion of ergodicity).

Notice (from step 1 in the construction of jump processes given in the previous subsection) that if the process is at state x, the process will tend to jump away from x rapidly if there is a lot of probability mass is contained in the neighborhood around x. If there is little probability in the surrounding neighborhood, the process will tend to spend more time loitering at x.

This is convenient since in implementation, there is no need for the computer to wait physically an amount of time w. When computing statistics from the chain, we can simply weight the samples according to the w's.

7. Sampling the Space of Blobs

Let $\mathcal{X}$ be the space of blobs, which are simply connected shapes on the 2-D lattice. We now consider different ways to construct MCMC-style algorithms that sample from the space of blobs. Using the definitions from Section 2, define the sets of blobs that can be reached via the addition (birth) and deletion (death) of a point along the boundary:

$$\mathcal{N}_b(x) = \{x \vee \delta(p) : p \in \mathcal{P}, aa(x)(p) = 1\}, \tag{28}$$

$$\mathcal{N}_d(x) = \{x - \delta(p) : p \in \mathcal{P}, ad(x)(p) = 1\}. \tag{29}$$

Let $\mathcal{N}(x) = \mathcal{N}_b(x) \cup \mathcal{N}_d(x)$. In the following examples, $\mathcal{N}(x)$ will be the set of states than can be reached from x in one step of the sampling algorithm.

7.1. Metropolis-Hasting Approach

One approach to sampling from the space of blobs is to construct a discrete-time Metropolis-Hastings sampler, where $\mathcal{N}(x)$ is the set of states that can be reached from x in one Metropolis-Hastings move. Specification of a Metropolis-Hastings algorithm requires choosing a *proposal density* $r(x_{old}, x_{prop})$ (a density in x_{prop} parameterized by x_{old}). Let π denote the desired density. At each step, the algorithm draws a

proposal x_{prop} from the proposal density. It accepts the proposal with *acceptance probability*

$$\alpha(x_{old}, x_{prop}) = \min\left\{\frac{\pi(x_{prop})r(x_{prop}, x_{old})}{\pi(x_{old})r(x_{old}, x_{prop})}, 1\right\}. \tag{30}$$

If the proposal is rejected, the algorithm remains at x_{old}. The generated chain $x(0), x(1), x(2), \ldots$ has the property that $x(n)$ converges (in an appropriate sense) to a random variable specified by the density π. There are many variations on this theme, including "deterministic scan" algorithms, which cycle through different proposal densities for different variables or blocks of variables in a nonrandom fashion. Consult Besag and Green, Smith and Roberts, W. Gilks (1993, 1993, 1996) for technical details about necessary conditions and modes of convergence.

Here are two possible algorithms (there are many others):

1) An easy proposer: $r(x_{old}, x_{prop}) =$

$$\frac{1}{|\mathcal{N}(x_{old})|} \text{for } x_{prop} \in \mathcal{N}(x_{old}),\ 0 \text{ otherwise} \tag{31}$$

$$\alpha(x_{old}, x_{prop}) = \min\left\{\frac{\pi(x_{prop})|\mathcal{N}(x_{old})|}{\pi(x_{old})|\mathcal{N}(x_{prop})|}, 1\right\}. \tag{32}$$

2) A smart proposer: $r(x_{old}, x_{prop}) =$

$$\frac{\pi(x_{prop})}{\sum_{x' \in \mathcal{N}(x_{old})} \pi(x')} \text{for } x_{prop} \in \mathcal{N}(x_{old}),\ 0 \text{ otherwise,} \tag{33}$$

$$\alpha(x_{old}, x_{prop}) = \min\left\{\frac{\sum_{x' \in \mathcal{N}(x_{old})} \pi(x')}{\sum_{x'' \in \mathcal{N}(x_{prop})} \pi(x'')}, 1\right\}. \tag{34}$$

The first algorithm proposes uniformly. Each proposal involves little computation, and a simple likelihood comparison is all that is needed for the acceptance probability. The second examines the likelihood of each move, and selects based on that. Although each proposal (and acceptance calculation) involves substantially more computation than the uniform selection, each individual proposal is more likely, yielding more powerful moves and a higher acceptance rate. Our initial experiments explored the first algorithm. We found it to be excruciatingly slow, which motivated the formulation of the second.

7.2. Continuous-Time Conditional-Gibbs Approach

Now we consider an approach based on the continuous-time algorithm proposed in Section 6.2. The space here is discrete, so the formulation of Section 6.2 can be applied with respect to an underlying counting measure, and our measure-theoretic integral notation simplifies to sums. Now, the amount of time to wait after we jump to state x is exponential with mean

$$\frac{1}{\sum_{z \in \mathcal{N}(x)} \pi(z)}. \tag{35}$$

After this time has passed, we draw a new state y from

$$\frac{\pi(y)}{\sum_{z \in \mathcal{N}(x)} \pi(z)} I_{\mathcal{N}(x)}(y). \tag{36}$$

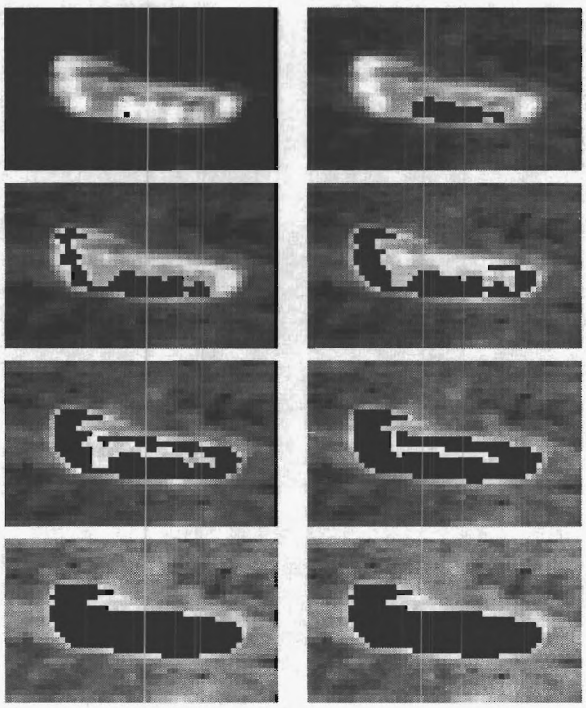


Figure 6. Generic shape inference on a tank provided by AMCOM, starting from a shape consisting of a single point in a hot region. The algorithm adds pixels to fill out the tank body.

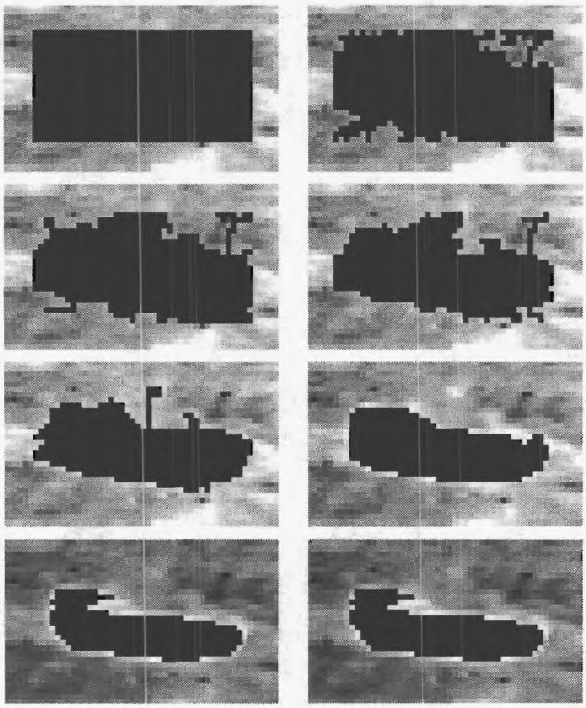


Figure 7. Generic shape inference on a tank, starting with a rectangle larger than the tank. The algorithm removes pixels until the tank body is left.

Example 1: To illustrate the behavior of the sampler, examples using the M60 from AMCOM (displayed in Figure 5) are shown in Figures 6 and 7, beginning with a single point and a rectangle that is much larger than the target, respectively. Snapshots of the underlying Markov chain are displayed. In the first example, the point, which is on a bright section of the tank, snakes along the outside, filling in the bright regions first, and then moving in to fill the less-bright central area. In the second example, the block decays to isolate the tank. The third row in the left column is particularly intriguing; notice there is a bright spot above the tank that the algorithm is loathe to disengage. After it pulls in the remainder of the background, it finally relents and withdraws the tentacle.

Example 2: A different example running on an M60 image provided by Dr. James Ratches of the U.S. Army Night Vision and Electronic Sensors Directorate (NVESD), shown in Figure 8, is illustrated in Figures 9 (showing iterations 100 to 800 of the underlying chain, at 100 step

Figure 8. FLIR image of an M60 (facing away from the detector) provided by NVESD.

increments) and 10 (showing iterations 1400 to 2800 of the underlying chain, at 200 step increments). Figure 9 demonstrates "burn-in period" behavior, while in Figure 10, the chain reaches equilibrium.

The upper left panel of Figure 11 shows the likelihood of the evolving blob, varying with time of the underlying discrete-time chain. Notice that it increases during the burn-in period and levels off in the equilibrium period. The lower left panel shows the corresponding total jump intensities, given by (35), associated with each time step. In computing averages, functionals of samples from the chain should be weighted inversely proportional to the jump intensity. Notice that during the burn-in period, the jump intensity is high, so the associated jump process will tend to jump quickly, while during the equilibrium period, the jump intensity is lower, so the process will linger longer at each state. The right panels of Figure 11 show versions of the left panels zoomed to show the equilibrium period in detail.

Summary statistics, computed from the weighted average of states from time 2000 to 3000, are illustrated in Figure 12. The top left panel shows, in gray scale, the percentage of time that a particular pixel is "on." In the top right panel, the grey pixels are pixels that were never on throughout that part of the process, white corresponds to the pixels that were on more than 50% of the time, and black corresponds to the remainder. The bottom left panel shows the pixels that were on more than 50% of the time superimposed as a black shape over the data. Taking the pixels that were on more than 50% of the time as an estimate of the blob shape, the bottom right panel shows the pixelwise difference between a synthesized scene consisting of only blob and background (with the appropriate estimated intensities for the foreground and the

background) and the data. Notice that the intensity of the blob is too low to accurately represent the intensity of the exhaust, while it is too high to accurately represent the region immediately surrounding the exhaust.

Of course, if these targets are present in the algorithm's 3-D target library, the rigid targets would be more preferable than the blobs. The blobs earn their keep when they accommodate targets not present in the library.

8. Limiting Cases of Discrete-State Continuous-Time Processes

We now present two theorems illustrating how continuous-time random processes defined on discretizations of $\Re$ with step size ϵ converge to diffusions as the discretization is made finer. The first theorem explores a special case of the continuous-time jumping algorithm outlined in Section 6.2. The second is analogous to a Metropolis-Hastings acceptance/rejection scheme. In the second case, the limiting diffusion is a Langevin diffusion. In the first case, the limiting diffusion is a different sort of a diffusion. We are not aware if it has been previously studied.

One advantage of the generator approach used in this section is that one can easily combine the small-step diffusions presented here with jumping processes that move between subspaces by adding the generators associated with the diffusion and jump processes as described in Amit et al. (1993). An elegant example of this approach is given in Section 5.4 of Srivastava (1996).

8.1. Conditional-Gibbs Style

THEOREM 6. *Let π be a strictly positive, twice-differentiable probability density on $\Re$ with a Gibbs representation $\pi(x) = \exp[E(x)]/Z$. Consider a family of jump processes $X_\epsilon(t)$, indexed by $\epsilon > 0$, with jump measures given by*

$$q_\epsilon(x, dy) = \frac{1}{\epsilon^2}\{\pi(x+\epsilon)\delta(y-(x+\epsilon)) + \pi(x-\epsilon)\delta(y-(x-\epsilon))\}\, dy, \tag{37}$$

where δ is the Dirac delta function. As $\epsilon \to 0$, X_ϵ converges in the Skorohod topology to a diffusion process X defined by the SDE

$$dX(t) = 2E'(X(t))\pi(X(t))dt + \sqrt{2\pi(X(t))}dW(t). \tag{38}$$

Moreover, $\pi(x)dx$ is a stationary measure for X.

Remark: Notice that (38) is somewhat similar to a Langevin SDE, with some important distinctions. The process still drifts along the gradient of the posterior, but the presence of the $\pi(X(t))$ in the drift term implies that the gradient pulls harder in exploring regions of high probability. Similarly, the Wiener process is more "turbulent" in regions on high probability. We have been unable to find (38) in the literature, and therefore do not have a name for it.

T. he generator for a jump process is

$$Af(x) = -q(x)f(x) + q(x)\int Q(x, dy)f(y). \tag{39}$$

Here, the jump intensity is

$$q_\epsilon(x) = \int q(x, dy) = \frac{1}{\epsilon^2}\left\{\pi(x+\epsilon) + \pi(x-\epsilon)\right\}. \tag{40}$$

The generator associated with index ϵ is given by

$$\begin{aligned} A_\epsilon f(x) \;=\; & \frac{1}{\epsilon^2}\left\{-[\pi(x+\epsilon) + \pi(x-\epsilon)]f(x)\right. \\ & \left. +\pi(x+\epsilon)f(x+\epsilon) + \pi(x-\epsilon)f(x-\epsilon)\right\}. \end{aligned} \tag{41}$$

Since we will be taking limits to form derivatives, we restrict the domain of f to be twice-differential bounded functions.

Employing Taylor series with terms up to the second order, the limit of (42) as $\epsilon \to 0$ is $\lim_{\epsilon\to 0} A_\epsilon f(x) =$

$$\begin{aligned} & \lim_{\epsilon\to 0}\frac{1}{\epsilon^2}\{-[\pi(x+\epsilon) + \pi(x-\epsilon)]f(x) \\ & +\pi(x+\epsilon)f(x+\epsilon) + \pi(x-\epsilon)f(x-\epsilon)\} \\ = \; & \lim_{\epsilon\to 0}\frac{1}{\epsilon^2}\{-[2\pi(x) + \epsilon^2\pi''(x)]f(x) \\ & +2f(x)\pi(x) + \epsilon^2(\pi\cdot f)''(x)\} \\ = \; & -\pi''(x)f(x) + \pi''(x)f(x) + 2\pi'(x)f'(x) + \pi(x)f''(x) \\ = \; & 2\pi'(x)f'(x) + \pi(x)f''(x), \end{aligned} \tag{42}$$

which is the generator of a diffusion with infinitesimal mean $2\pi'(x) = 2E'(x)\pi(x)$ and infinitesimal variance $2\pi(x)$ (Ethier and Kurtz (1986), p. 366, Eq. 1.2). This process can be implicitly defined by the SDE

$$dX(t) = 2E'(X(t))\pi(X(t))dt + \sqrt{2\pi(X(t))}dW(t), \tag{43}$$

where $W(t)$ is a standard Wiener process.

If we take, for instance, $\epsilon = 1/n$, then applying Theorems 4 and 5 reveals that $X_\epsilon \to X$ in the Skorohod topology as $\epsilon \to 0$.

By Theorem 2, π is a stationary density of X if and only if $\int A_0 f(x)\pi(x)dx = 0$ for all f in the domain of the generator:

$$\int A_0 f(x)\pi(x)dx = \int \pi^2(x)f''(x)dx + 2\int \pi'(x)\pi(x)f'(x)dx. \quad (44)$$

Applying integration by parts (let $u = \pi^2(x)$ and $dv = f''(x)dx$, so $du = 2\pi(x)\pi'(x)dx$ and $v = f'(x)$) to the first term and employing the fact that $\pi(x)$ and $f'(x)$ must both go to zero at the extremes yields

$$\begin{aligned} & \int A_0 f(x)\pi(x)dx \\ &= \pi^2(x)f'(x)|_{-\infty}^{\infty} - 2\int \pi'(x)\pi(x)f'(x)dx + 2\int \pi'(x)\pi(x)f'(x)dx \\ &= 0, \end{aligned} \quad (45)$$

which completes the proof.

8.2. Metropolis Style

An informal recollection of the classic construction of Brownian motion in $\Re$ in terms of coin flips will be helpful in interpreting the next theorem. At each time step, if the coin comes up heads, our Brownian traveller takes a step to the left; if tails, a step to the right. Letting the time between steps and the size of the steps shrink yields a Brownian motion. Now, suppose we place a density π over the range of places the traveller may sojourn, and suppose that instead of automatically taking a step after each coin flip, the traveller first looks at the value of the density π at the place he is asked to move and compares it with the value of π where he is currently at. If the probability is higher, he takes the step; if lower, he decides to accept or reject the step in the traditional Metropolis fashion. This heuristic scenario is made precise in the next theorem, where we find that this alternate traveller's journey, instead of converging to Brownian motion, remarkably converges to a Langevin diffusion with stationary density π.

THEOREM 7. *Let π be a strictly positive, twice-differentiable probability density on $\Re$ with a Gibbs representation $\pi(x) = \exp[E(x)]/Z$. Consider a family of jump processes $X_\epsilon(t)$, indexed by $\epsilon > 0$, with jump measures given by $q_\epsilon(x, dy) =$*

$$\begin{aligned} & \frac{1}{\epsilon^2}\left\{\frac{1}{2}\min\left[\frac{\pi(x+\epsilon)}{\pi(x)}, 1\right]\delta(y-(x+\epsilon))\right. \\ + \; & \left.\frac{1}{2}\min\left[\frac{\pi(x-\epsilon)}{\pi(x)}, 1\right]\delta(y-(x-\epsilon))\right\} dy. \end{aligned} \quad (46)$$

As $\epsilon \to 0$, X_ϵ converges in the Skorohod topology to a diffusion process X defined by the Langevin SDE

$$dX(t) = \frac{1}{2}E'(X(t))dt + dW(t). \tag{47}$$

Moreover, $\pi(x)dx$ is a stationary measure for X.

T. he jump intensity is

$$q_\epsilon(x) = \int q(x, dy) = \frac{1}{2\epsilon^2}\left\{\min\left[\frac{\pi(x+\epsilon)}{\pi(x)}, 1\right] + \min\left[\frac{\pi(x-\epsilon)}{\pi(x)}, 1\right]\right\}. \tag{48}$$

The generator associated with index ϵ is given by $A_\epsilon f(x) =$

$$\frac{1}{2\epsilon^2}\left\{-\left[\min\left(\frac{\pi(x+\epsilon)}{\pi(x)}, 1\right) + \min\left(\frac{\pi(x-\epsilon)}{\pi(x)}, 1\right)\right] f(x) \right.$$
$$\left. + \min\left(\frac{\pi(x+\epsilon)}{\pi(x)}, 1\right) f(x+\epsilon) + \min\left(\frac{\pi(x-\epsilon)}{\pi(x)}, 1\right) f(x-\epsilon)\right\} \tag{49}$$

It will be easiest to consider separate cases. For $\pi(x+\epsilon) > \pi(x)$, $\pi(x-\epsilon) < \pi(x)$, the generator (49) simplifies to

$$A_\epsilon f(x) = \frac{1}{2\epsilon^2}\left\{-\left[1 + \frac{\pi(x-\epsilon)}{\pi(x)}\right] f(x) + f(x+\epsilon) + \frac{\pi(x-\epsilon)}{\pi(x)} f(x-\epsilon)\right\}. \tag{50}$$

Expanding functions with ϵ arguments in Taylor series to the second order yields $\lim_{\epsilon\to 0} A_\epsilon f(x) =$

$$\lim_{\epsilon\to 0}\frac{1}{2\epsilon^2}\left\{-\left[1 + \frac{\pi(x) - \epsilon\pi'(x) + \epsilon^2\pi''(x)}{\pi(x)}\right] f(x) \right.$$
$$+ f(x) + \epsilon f'(x) + \epsilon^2 f''(x)$$
$$\left. + \frac{\pi(x)f(x) - \epsilon(\pi\cdot f)'(x) + \epsilon^2(\pi\cdot f)''(x)}{\pi(x)}\right\}$$
$$= \lim_{\epsilon\to 0}\frac{1}{2\epsilon^2}\left\{-\left[1 + \frac{\pi(x) - \epsilon\pi'(x) + (\epsilon^2/2)\pi''(x)}{\pi(x)}\right] f(x) \right.$$
$$+ f(x) + \epsilon f'(x) + \frac{\epsilon^2}{2} f''(x)$$
$$+ \frac{\pi(x)f(x) - \epsilon[\pi'(x)f(x) + \pi(x)f'(x)]}{\pi(x)}$$
$$\left. + \epsilon^2\frac{\pi''(x)f(x) + 2\pi'(x)f'(x) + \pi(x)f''(x)}{2\pi(x)}\right\}$$
$$= \lim_{\epsilon\to 0}\frac{1}{2\epsilon^2}\left\{-2f(x) + \epsilon\frac{\pi'(x)f(x)}{\pi(x)} - \epsilon^2\frac{\pi''(x)f(x)}{2\pi(x)}\right.$$

$$+2f(x) - \epsilon\frac{\pi'(x)f(x)}{\pi(x)} + \epsilon^2\frac{\pi''(x)f(x)}{2\pi(x)}$$
$$+\epsilon^2\frac{\pi'(x)f'(x)}{\pi(x)} + \epsilon^2 f''(x)\Big\}$$
$$= \frac{\pi'(x)}{2\pi(x)}f'(x) + \frac{1}{2}f''(x). \tag{51}$$

Working this through for the opposite case, $\pi(x+\epsilon) < \pi(x)$, $\pi(x-\epsilon) > \pi(x)$, yields the same result. Now consider the case where $\pi(x+\epsilon) < \pi(x)$, $\pi(x-\epsilon) < \pi(x)$, and suppose this case holds as $\epsilon \to 0$ so that $\pi'(x) = 0$. Here $\lim_{\epsilon\to 0} A_\epsilon f(x) =$

$$\lim_{\epsilon\to 0}\frac{1}{2\epsilon^2}\left\{-\frac{2\pi(x)+\epsilon^2\pi''(x)}{\pi(x)}f(x) + \frac{2\pi(x)f(x) + \epsilon^2(\pi\cdot f)''(x)}{\pi(x)}\right\}$$
$$= \frac{1}{2}\left\{-\frac{\pi''(x)}{\pi(x)}f(x) + \frac{\pi''(x)f(x)}{\pi(x)} + 2\pi'(x)f'(x) + \frac{\pi(x)f''(x)}{\pi(x)}\right\}$$
$$= \frac{f''(x)}{2}. \tag{52}$$

Finally, consider the case where $\pi(x+\epsilon) > \pi(x)$, $\pi(x-\epsilon) > \pi(x)$, and again suppose this case holds as $\epsilon \to 0$ so $\pi'(x) = 0$. Now we have:

$$\begin{aligned}\lim_{\epsilon\to 0} A_\epsilon f(x) &= \lim_{\epsilon\to 0}\frac{1}{2\epsilon^2}\{-2f(x) + f(x+\epsilon) + f(x-\epsilon)\}\\ &= \lim_{\epsilon\to 0}\frac{1}{2\epsilon^2}\left\{-2f(x) + 2f(x) + \epsilon^2 f''(x)\right\}\\ &= \frac{f''(x)}{2}.\end{aligned} \tag{53}$$

Notice that (52) and (53) can both be written as (51), which is the generator of a diffusion process with infinitesimal mean $\pi'(x)/[2\pi(x)] = E'(x)/2$ and infinitesimal variance 1. This process can be implicitly defined by the SDE for the Langevin diffusion

$$dX(t) = \frac{1}{2}E'(X(t))dt + dW(t), \tag{54}$$

where $W(t)$ is a standard Wiener process. As in the preceding theorem, applying Theorems 4 and 5 reveals that $X_\epsilon \to X$ in the Skorohod topology as $\epsilon \to 0$.

Again by Theorem 2, π is stationary density of X if and only if $\int A_0 f(x)\pi(x)dx = 0$ for all f in the domain of the generator:

$$\int A_0 f(x)\pi(x)dx = \frac{1}{2}\left\{\int \pi(x)f''(x)dx + \int \pi'(x)f'(x)dx\right\}. \tag{55}$$

Applying integration by parts (let $u = \pi(x)$ and $dv = f''(x)dx$, so $du = \pi'(x)dx$ and $v = f'(x)$) to the first term and employing the fact that $\pi(x)$ and $f'(x)$ must both go to zero at the extremes yields $\int A_0 f(x)\pi(x)dx =$

$$\frac{1}{2}\left\{\pi(x)f'(x)|_{-\infty}^{\infty} - \int \pi'(x)f'(x)dx + \int \pi'(x)f'(x)dx\right\}, \qquad (56)$$

which is zero.

Remark: Consult Section 3.4 of Grenander and Miller (1991) and Section 5.5 of Srivastava (1996) for discussions on the ergodic properties of diffusion processes, including the uniqueness of their invariant measures and their convergence to those invariant measures.

9. Conclusions

In the literature, unstructured representations (i.e. segmentations) are generally intended as intermediate steps on the way to structured representations (target types). In our Grenander-inspired framework, unstructured and structured representations are intended to coexist within the same scene. In particular, the blobs of Section 2 offer a novel way of allowing the algorithm to explain interesting structures in the scene that may not be present in the target library. The main point is not that our blobs are the *only* feasible representation, but that introducing *some* kind of unstructured representation is essential if ATR algorithms are to be robust to "clutter" not present in their target library. We have favored blobs due to their simplicity and convenience.

This paper presented two primary contributions along this path:

1. Deterministic region growing algorithms are ubiquitous in the literature. Section 7 presented a stochastic algorithm in which blobs grow and shrink probabilistically. In the literature, stochastic algorithms have been applied to operate on images on a pixel-by-pixel basis, where each pixel has a label (Geman and Geman, 1984); the product of the set of labels for each pixel forms the parameter space, and priors such as the celebrated Ising model (Kinderman and Snell, 1980) are used to encourage clumping. Here, in contrast, the *blob itself is the parameter*. Evolving simply connected shapes is then straightforward.

2. Section 8 presented two new theorems, which demonstrate that certain random sampling algorithms operating on a discretized space,

in which moves are restricted to neighboring points, converge to diffusion algorithms as the discretization is refined to the continuum.

For simplicity, this paper considered a single blob against a background. This will need to be extended to sampling the space of multiple blobs. The more difficult task will be developing appropriate jumping schemes for moving between blob and rigid-target representations.

Acknowledgements

The author would like to thank Profs. Donald Snyder, Michael Miller, Joseph O'Sullivan, Daniel Fuhrmann, and Bijoy Ghosh for comments on the thesis chapters that formed the seed of these investigations, Prof. Ulf Grenander for general guru guidance and the encouragement to go ahead and use the word "blobs," and Dr. Jon Sjogren of AFOSR for supporting the continuation of this work.

References

Amit, Y., U. Grenander, and M. Miller: January 1993, 'Ergodic Properties of Jump-Diffusion Processes'. *Monograph of the Electronic Signals and Systems Research Laboratory.*

Barron, A., J. Rissanen, and B. Yu: 1998, 'THe Minimum Description Length Principle in Coding and Modeling'. *IEEE Trans. on Information Theory* **44**(6), 2743–2760.

Besag, J.: 1994, 'Contribution to the Discussion of the Paper by Grenander and Miller'. *Journal of the Royal Statistical Society B* **56**(3), 591–592.

Besag, J. and P. J. Green: 1993, 'Spatial statistics and Bayesian computation'. *J. Royal Statistical Society B* **55**(1), 25–38.

Durrett, R.: 1996, *Stochastic Calculus: A Practical Introduction.* Boca Raton: CRC Press.

Ethier, S. and T. Kurtz: 1986, *Markov Processes: Characterization and Convergence.* John Wiley and Sons.

Figueiredo, M. A. T., J. M. N. Leitao, and A. K. Jain: 1997, 'Adaptive Parametrically Deformable Contours'. In: M. Pellilo and E. Hancock (eds.): *Energy Minimization Methods in Computer Vision and Pattern Recognition.* Springer-Verlag, pp. 35–50.

Gelfand, S. and S. Mitter: 1991, 'Weak Convergence of Markov Chain Sampling Methods and Annealing Algorithms to Diffusions'. *Journal of Optimization Theory and Applications* **68**(3), 483–498.

Geman, S. and D. Geman: 1984, 'Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images'. *IEEE Trans. Pattern Anal. Machine Int.* **6**, 721–741.

Geman, S. and C.-R. Hwang: 1987, 'Diffusions for Global Optimization'. *SIAM J. Control and Optimization* **24**, 1031–1043.

Gillespie, D.: 1992, *Markov Processes: An Introduction for Physical Scientists.* San Diego: Academic Press.

Grenander, U.: 1994, *General Pattern Theory: A Mathematical Study of Regular Structures.* Oxford Univ. Press.

Grenander, U.: 1996, *Elements of Pattern Theory.* Johns Hopkins Univ. Press.

Grenander, U., Y. Chow, and D. Keenan: 1990, *HANDS: A Pattern Theoretic Study of Biological Shapes.* New York: Springer-Verlag.

Grenander, U. and D. Keenan: 1993, 'On the Shape of Plane Images'. *SIAM J. Appl. Math.* **53**(4), 1072–1094.

Grenander, U. and K. Manbeck: 1993, 'A Stochastic Shape and Color Model for Defect Detection in Potatoes'. *Journal of the American Statistical Association* **2**(2), 131–151.

Grenander, U. and M. I. Miller: 1991, 'Jump-Diffusion Processes for Abduction and Recognition of Biological Shapes'. *Monograph of the Electronic Signals and Systems Research Laboratory.*

Grenander, U. and M. I. Miller: 1994, 'Representations of Knowledge in Complex Systems'. *Journal of the Royal Statistical Society B* **56**(3), 549–603.

Kass, M., A. Witkin, and D. Terzopolous: 1988, 'Snakes: Active contour models'. *International Journal of Computer Vision* **1**(4), 321–331.

Kichenessamy, S., P. Olver, A. Tannenbaum, and A. Yezzi: 1997, 'Geometric Active Contours for Segmentation of Medical Imagery'. *IEEE Trans. on Medical Imaging* **16**, 199–209.

Kinderman, R. and J. L. Snell: 1980, *Markov Random Fields and their Applications.* American Mathematical Society.

Lanterman, A.: 1998, *Modeling Clutter and Target Signatures for Pattern-Theoretic Understanding of Infrared Scenes.* St. Louis, MO: D.Sc. Dissertation, Dept. of Electrical Engineering, Sever Institute of Technology, Washington Univ.

Lanterman, A.: 2000, 'Bayesian Inference of Thermodynamic State Incorporating Schwarz-Rissanen Complexity for Infrared Target Recognition'. *Optical Engineering* **39**(5), 1282–1292.

Lanterman, A.: 2001, 'Schwarz, Wallace, and Rissanen: Intertwining Themes in Theories of Model Order Estimation'. *International Statistical Review* **69**(2), 185–212.

Lanterman, A., M. Miller, and D. Snyder: 1997a, 'General Metropolis-Hastings jump diffusions for automatic target recognition in infrared scenes'. *Optical Engineering* **36**(4), 1123–1137.

Lanterman, A., M. Miller, and D. Snyder: 1997b, 'Representations of Shape for Structural Inference in Infrared Scenes'. In: F. Sadjadi (ed.): *Automatic Object Recognition VII*, Vol. SPIE Proc. 3069. Orlando, FL, pp. 257–268.

Leclerc, Y.: 1989, 'Constructing Simple Stable Descriptions for Image Partitioning'. *International Journal of Computer Vision* **3**(1), 73–102.

Miller, M., U. Grenander, J. A. O'Sullivan, and D. Snyder: 1997, 'Automatic Target Recognition Organized via Jump-Diffusion Algorithms'. *IEEE Trans. on Image Processing* **6**(1), 1–17.

Miller, M., A. Srivastava, and U. Grenander: 1995, 'Conditional-Mean Estimation Via Jump-Diffusion Processes in Multiple Target Tracking/Recognition'. *IEEE Trans. on Signal Processing* **43**(11), 2678–2690.

Reno, A. and D. Booth: 1998, 'Deformable models for object recognition in aerial images'. In: F. Sadjadi (ed.): *Automatic Target Recognition VIII*, Vol. SPIE Proc. 3371. Orlando, FL. 322-333.

Roberts, G. and J. Rosenthal: 1998, 'Optimal Scaling of Discrete Approximations to Langevin diffusions'. *Journal of the Royal Statistical Society B* **60**(1), 255–268.

Sethian, J. A.: 1996, *Level Set Methods and Fast Marching Methods.* Cambridge University Press.

Smith, A. and G. Roberts: 1993, 'Bayesian Computation via the Gibbs Sampler and Related Markov Chain Monte Carlo Methods'. *J. Royal Statistical Society B* **55**(1), 3–37.

Snyder, D., A. Hammoud, and R. White: 1993, 'Image recovery from data acquired with a charge-coupled-device camera'. *Journal of the Optical Society of America A* **10**(5), 1014–1023.

Snyder, D. L. and M. I. Miller: 1991, *Random Point Processes in Time and Space.* Springer-Verlag, 2nd edition.

Srivastava, A.: 1996, *Inference on Transformation Groups Generating Patterns on Rigid Motions.* St. Louis, MO: D.Sc. Dissertation, Dept. of Electrical Engineering, Sever Institute of Technology, Washington Univ.

W. Gilks, S. Richardson, D. S. (ed.): 1996, *Markov Chain Monte Carlo in Practice.* Chapman and Hall.

Zhu, S. and A. Yuille: 1996, 'Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multiband Image Segmentation'. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **18**(9), 884–900.

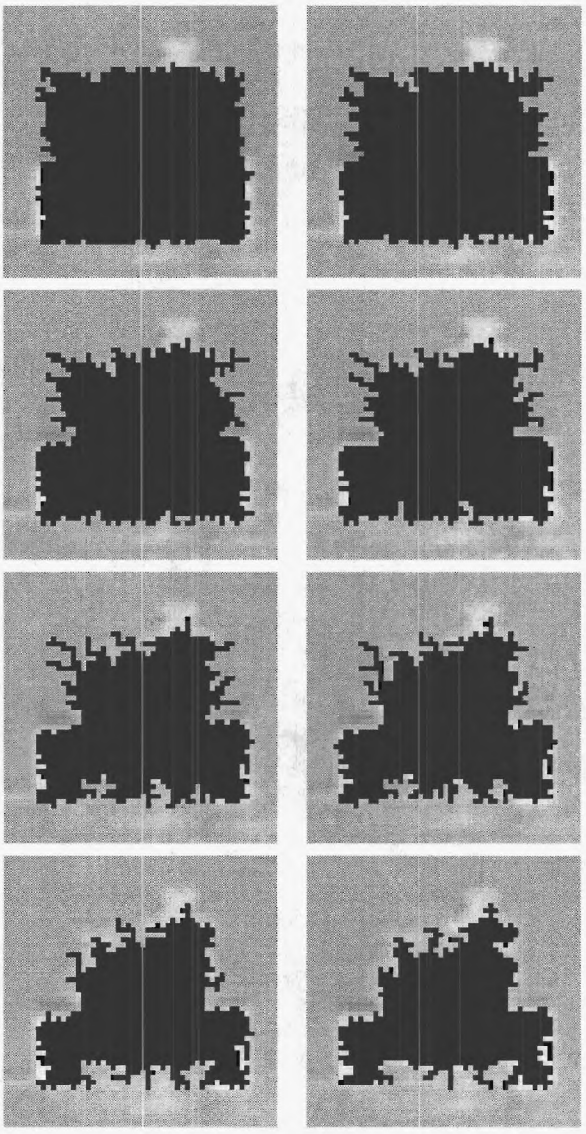


Figure 9. Generic shape inference on a tank provided by NVESD. Panels show snapshots of underlying chain from iteration 100 to 800 at 100 step increments.

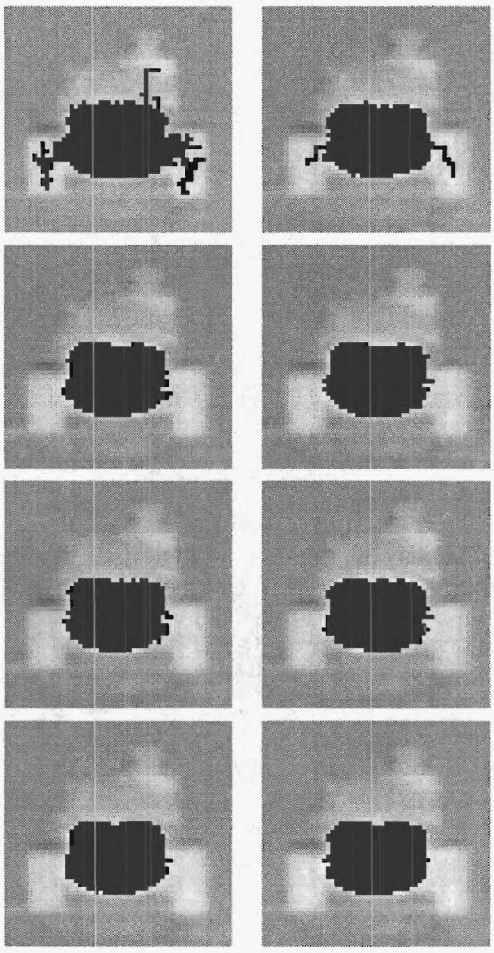


Figure 10. Generic shape inference on a tank provided by NVESD. Panels show snapshots of underlying chain from iteration 1400 to 2800 at 200 step increments.

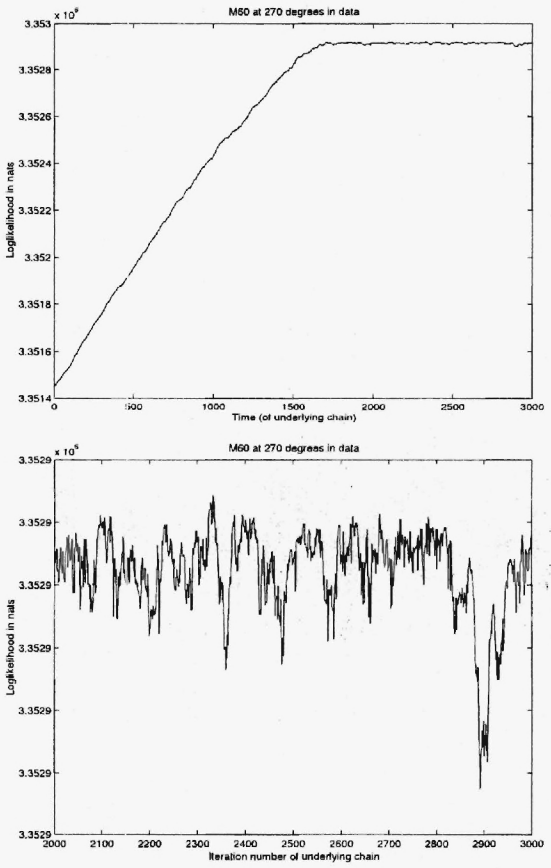


Figure 11. Upper panel shows loglikelihood with respect to time of the underlying chain; lower panel shows associated jump intensity.

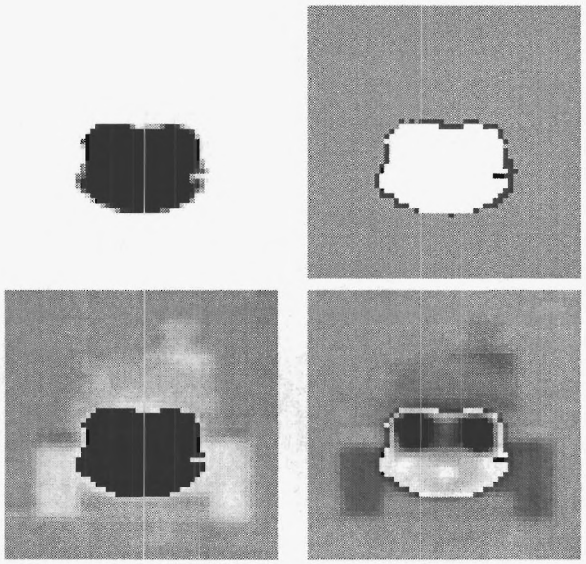


Figure 12. Top left panel: percentage of time each pixel is considered part of the blob, shown via greyscale. Top right panel: Grey, pixel never considered "on"; white, on more than 50% of the time; black otherwise. Bottom left panel: White part of Panel 2, superimposed over data. Bottom right panel: Simulated scene consisting only of the blob from Panel 3 and the background, minus the data (pixelwise difference).

Appendix G

M. Tobias and A.D. Lanterman, "Techniques for Birth Particle Placement in the PHD Particle Filter, Applied to Passive Radar," *IEE Proc. Radar, Sonar, and Navigation*, submitted April 7, 2007.

Techniques for Birth Particle Placement in the PHD Particle Filter Applied to Passive Radar

Martin Tobias[†] and Aaron D. Lanterman
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250 U.S.A.
E-mail: mtobias@ll.mit.edu, lanterma@ece.gatech.edu

Abstract

An initial investigation found Ronald Mahler's Probability Hypothesis Density (PHD) filter to be a promising tool for the passive coherent location (PCL) of targets observed via multiple bistatic radar measurements. This paper introduces two significant improvements to the particle-filter implementation of the PHD-based multitarget, multisensor tracker for PCL. The first is a simple, yet novel, peak-extraction technique that exploits the integral property of the PHD and provides a faster and more accurate method over currently used techniques to extract the target states from the PHD. The second is an improved method of placing birth particles, whereby a pre-computed range-variance-based grid and a least-squares iterative technique is used to place birth particles at the bistatic range and Doppler observation intersections. This leads to greater reliability in detecting targets and a significant reduction in the number of particles required for tracking.

I. Introduction

A particle-filter implementation of Mahler's Probability Hypothesis Density (PHD)-based Bayesian filter equations was applied in [1] to a multitarget, multisensor passive coherent location (PCL) scenario that used simulated range and Doppler observations. The PHD was represented by a collection of particles, each with state $\xi_i = [x\ y\ \dot{x}\ \dot{y}]$, that was propagated forward at each time step k according to a constant velocity model with Gaussian process

[†]Currently at M.I.T. Lincoln Laboratory

noise, and whose associated weights, w_i, were updated using the sensor observations in the following manner:

$$w_{i,k+1} = \left(\sum_{n=1}^{m} u_{i,n}\right) + \tilde{w}_{i,k+1}\left(1 - p_D(\xi)\right), \tag{1}$$

where

$$u_{i,n} = \frac{p_D(\xi_i) f(z_n|\xi_i)\tilde{w}_{i,k+1}}{\lambda c(z_n) + \sum_j p_D(\xi_j) f(z_n|\xi_j)\tilde{w}_{j,k+1}}, \tag{2}$$

m is the number of observations, p_D is the probability of detection, λ and $c(z)$ are the false-alarm parameters, $f(z|\xi)$ is the single-sensor likelihood that an observation z will be caused by a target with state ξ, and $\tilde{w}_{i,k+1}$ are the weights of both the propagated particles and any birth particles. The expected[1] number of targets present was obtained by summing the weights of the PHD:

$$\tilde{N} = E\left[no.\ of\ targets\right] = \left[\sum_i w_{i,k+1}\right]_{nearest\ integer.} \tag{3}$$

The PHD-based particle filter was found to be a promising tracker due to its ability to automatically estimate the number of targets present and to fuse easily different types of observation data to enhance tracking performance. However, because of the inherent sub-optimality of the particle filter, a clustering method of placing particles to model new targets was derived, whereby particles were placed according to a Gaussian distribution centered where the bistatic-range ellipse observations intersected the edges of the field of view (FoV). This method resulted in better tracking performance than when particles were placed around the FoV in a uniformly random manner. However, it was still neither efficient nor completely effective. Targets would often slip by undetected, and the number of particles needed was quite large. Section III of this paper presents a new technique that uses a pre-computable grid to place particles at the intersections of the bistatic-range ellipse and Doppler observations. This particle-placement method results in a significant improvement in target detection and a substantial reduction in the number of particles required.

Another improvement to the PHD-based particle implementation is presented in Section II. In the PHD-based particle filter, the particles found at the peak locations correspond to the target states, and a peak-extraction technique is needed to locate the peaks in the PHD. In [1], an expectation-maximization (EM) algorithm was used for the peak-extraction, whereby

[1]The term "expected" is used in the mathematical sense of the expected value of a random number, not in the sense of "supposed," "assumed," or "predicted." In the random set framework used to derive the PHD, the number of objects in the target-state set is a random variable.

a mixture of Gaussians was fitted to the PHD, where each Gaussian represented a target state. However, the EM algorithm was found to have high computational intensity and rather poor performance. Other particle-filter implementations [2]–[10] of the PHD filter have used similar Gaussian-mixture fitting techniques or clustering, such as k-means, to do the peak extraction. Section II introduces a new peak extraction technique that exploits the properties of the PHD to find the peaks. This new technique is found to extract the target states from the PHD much better than the EM algorithm and is comparable to, if not slightly better than, the k-means clustering algorithm.

The PHD particle filter implementation presented in this paper is a realistic multitarget tracking simulation modeling the passive radar configuration at the NATO Consultation, Command and Control Agency (NC3A) in The Hague, Netherlands. The equations for the signal to noise ratio (SNR), probability of detection (p_D), and other simulation parameters are as given in Sections 5.1-5.4 of [1]. The PHD particle filter implementation used is the same as in [1], except the birth particle placement and weighting are as presented in Section III.

II. Peak Extraction

New peak extraction logic was created to be used in lieu of the expectation-maximization (EM) algorithm to extract the target locations from the PHD. As noted in [1], the EM algorithm used to extract the peaks from the PHD exhibited undesirable behavior. Apart from having a considerably long execution time, the EM algorithm frequently failed to find the correct peak locations. The algorithm often aborted because it produced a singular covariance matrix or because it took too many iterations to fit the Gaussians to the PHD. Other times, it would "double-fit" Gaussians to a single peak, thereby extracting a single target twice. Essentially, the poor performance of the algorithm was due to our attempt at fitting a Gaussian mixture to a density that was not a Gaussian mixture.

Our new peak extraction algorithm is similar to the CLEAN technique used in astronomy to uncover secondary objects in an image by removing the effects of the primary objects [11]. The new peak extraction technique takes advantage of the properties of the PHD, and it works more accurately and much faster than the EM algorithm. It takes the highest peak in the PHD as the target location, and then extracts a target's worth of weight from the PHD at and around this point before searching for the next highest peak. Thus, the property of the PHD that it integrates to the expected number of targets is exploited. Pseudo-code for this

new peak extraction algorithm is provided in Table I.

Unlike the EM algorithm, the new algorithm is guaranteed to return a state value for every peak it is told to extract. This is because it simply returns the state of the particle with the largest weight when looking for a peak. It then constructs a neighborhood, based on the range and Doppler resolutions,[2] around that particle in the state space. The region of the neighborhood is increased until the sum of the weights of the particles inside it are equivalent to a target's worth of weight. The weight is then subtracted from the region – thereby, effectively extracting the target – and the procedure is repeated for the next desired peak. An illustration of the algorithm is given in Fig. 1. Occasionally, especially when the algorithm is extracting the final desired peak in a time step, the actual weight in the neighborhood around the peak is not sufficient to sum up to a target's worth. In this case, the algorithm still simply reports the target state of the peak and continues as normal.

The new peak-extraction algorithm was found to perform better, and significantly faster, than the EM algorithm. The new peak-extraction technique was also compared to a k-means clustering algorithm, and it provided results that were as good, if not better, than the k-means algorithm. The new peak-extraction algorithm performed better in the case where there were two peaks in the PHD but where only one target was estimated as being present. In this case, the k-means algorithm produced a target detection that was located somewhere between the two peaks; whereas, our peak extraction algorithm extracted the single target correctly.

[2]The range resolution is computed as $\Delta_R = \frac{c}{\beta_{min}}$, where c is the speed of light, and β_{min} is the smallest bandwidth of the transmitters used. The Doppler resolution is $\Delta_{\dot{R}} = \frac{\lambda_{max}}{CPI}$, where CPI is the coherent processing interval, and λ_{max} is the longest wavelength used among the transmitters.

TABLE I

PSEUDOCODE FOR PEAK EXTRACTION ALGORITHM

1) *Compute radius vector, $\boldsymbol{\rho}$, to use for determining region of peak extraction:*
   - $\boldsymbol{\rho} = [\Delta_R, \Delta_R, 2\Delta_{\dot{R}}, 2\Delta_{\dot{R}}]^{\mathsf{T}}$, where
   - $\Delta_R = \frac{c}{\beta_{min}}$ is the largest range resolution among the transmitters, and
   - $\Delta_{\dot{R}} = \frac{\lambda_{max}}{CPI}$ is the largest Doppler resolution (in m/sec) among the transmitters.
2) *Determine number of peaks to extract and calculate weight of each peak. The rounding error is assumed to be distributed evenly over all targets, and a 1% margin is added to the weight extraction:*
   - Expected number of peaks = round $(\sum w_{i,k+1})$.
   - Target weight = $\min\left(0.99,\ 0.99 \cdot \frac{\sum w_{i,k+1}}{\text{Expected number of peaks}}\right)$.
   - Let w_i be a copy of $w_{i,k+1}$, which are the particle weights in the PHD filter at time $k+1$.
     Note that the set of w_i weights are modified during the course of the peak extraction. The actual $w_{i,k+1}$ in the PHD filter are unaltered by the peak extraction algorithm.
3) *Extract peaks:* for p = 1 to Expected number of peaks,
   - Extracted peak = $\boldsymbol{\xi}_{j,k+1}$, where $j = \arg_i \max(w_i)$.
   - Let $w_{max} = w_j$.
   - Number of peaks found = Number of peaks found + 1.
   - If $w_j \geq$ Target weight, then set $w_j = w_j -$ Target weight.
     Return $\boldsymbol{\xi}_{j,k+1}$ as the extracted peak. Continue with the p-for-loop to find the next peak.
   - Else, for $n = 1$ to MaxTries,
     - Neighborhood = $\boldsymbol{\rho} \cdot n$.
     - Find all particles $\boldsymbol{\xi}_{\alpha,k+1}$ and their corresponding weights w_α, that are in the given neighborhood of $\boldsymbol{\xi}_{j,k+1}$. That is, find $(\boldsymbol{\xi}_{\alpha,k+1}, w_\alpha)$, such that:

       $$\boldsymbol{\xi}_{\alpha,k+1} \in [\boldsymbol{\xi}_{j,k+1} - n\boldsymbol{\rho},\ \boldsymbol{\xi}_{j,k+1} + n\boldsymbol{\rho}].$$

     - Neighborhood weight = $\sum_\alpha w_\alpha$. (Note that w_{max} is included in the Neighborhood weight.)
     - If Neighborhood weight $\geq$ Target weight, or $n =$ MaxTries, then a peak has been found or the current peak extraction is being cut short. In either case, reduce the weight of the particles in the Neighborhood by the Target weight:
       * Set $w_j = 0$, since $w_{max} \leq$ Target weight.
       * Set

         $$w_\alpha = w_\alpha \cdot \left(1 - \frac{\text{Target weight} - w_{max}}{\text{Neighborhood weight}}\right),$$

         such that the weight in the neighborhood is reduced by a target's amount of weight, not including the weight already removed at w_j. Note that the weight is removed proportionally, so that the peak structure in the particles is preserved.
       * Return $\boldsymbol{\xi}_{j,k+1}$ as the extracted peak. Continue with the p-for-loop to find the next peak.
     - Else, a target's amount of weight does not exist in the current neighborhood. Continue with the n-for-loop to expand the neighborhood.

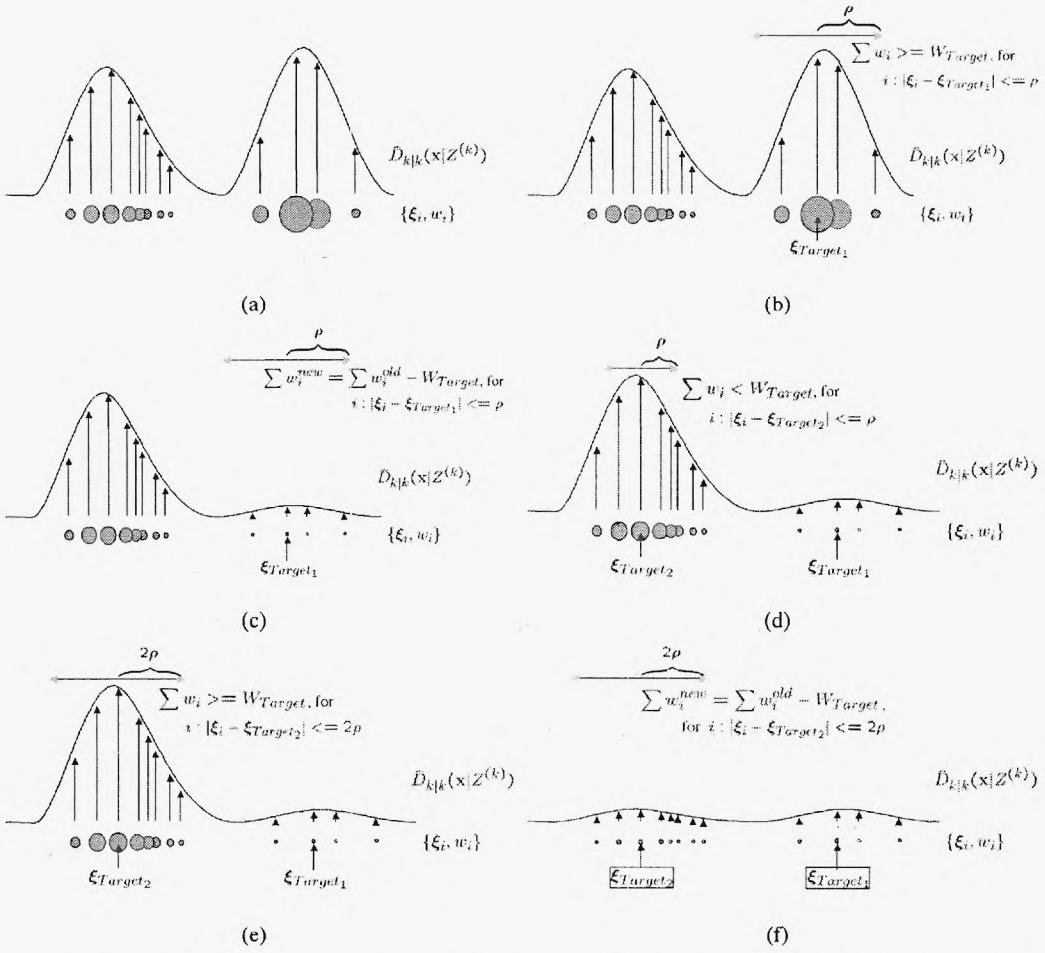


Fig. 1. An example of the peak extraction algorithm where the weights w_i of the particles ξ_i sum to two, as shown in Fig. 1(a), indicating the presence of two targets. The first step, shown in Fig. 1(b), is to find the particle with the greatest weight and take it to be the target state of Target One. Next, the weight within a radius of ρ, where we used $\rho = [\Delta_R, \Delta_R, 2\Delta_{\dot{R}}, 2\Delta_{\dot{R}}]^{\mathsf{T}}$, is integrated. In this example, the sum of the weights is greater than a target's worth, and so a target's worth of weight is proportionally subtracted from the particles within the radius, as shown in Fig. 1(c), and the algorithm proceeds to find the next target. The particle that now has the greatest weight is taken to be the state of Target Two, and the weights within a radius ρ are again summed up. This time, in Fig. 1(d), the total weight within the radius is less than a target's worth of weight, and so the radius is expanded until a target's worth is found. The algorithm thus continues in Figs. 1(e) and 1(f). Note that both targets sought have been assigned state values and that the algorithm could have stopped at Fig. 1(d). Note also that the peak extraction algorithm uses a copy of the PHD to perform these steps and that the weights and particles representing the actual PHD propagated in the Bayesian PHD particle filter have not been touched.

III. Improved Birth Particle Placement

In the PHD particle-filter tracker implementation in [1], the birth particles are distributed either uniformly around the edges of the FoV or in a clustered placement at the points where the bistatic ellipses intersect the edges of the FoV. However, as suggested in the results and conclusions of that paper, the birth particle placement used in [1] needs to be replaced with a smarter proposal function, since it often fails to detect the targets. A smarter proposal for the birth particles is described below, in which birth particles are placed at the intersections of the bistatic range ellipse observations. As part of this improvement, the logic for placing birth particles around the edge of the FoV, either uniformly when no ellipse intersections are present or in a clustered fashion when ellipses are present, was removed. This change also avoided the target overestimation problem, described in the conclusions of [1], caused by birth particles being placed in areas of low SNR.

Two techniques for implementing the smarter birth particle placement are considered. The first focuses on range resolution, and the second focuses on range variance. We found that the second technique worked better.

A. The Range-Resolution Based Grid Technique

In an attempt to more cleverly distribute birth particles, ellipse-intersection-counting grid logic, based on range resolution, and recursive, combinatorial, Doppler-intersection finding logic were added to the multitarget tracker. This logic localizes the placement of the birth particles to the intersections of the bistatic range ellipses and initializes the birth particles with appropriate velocity parameters.

The counting-grid on which to localize the ellipse intersections is constructed by forming overlapping gridspaces, each of which has a width equivalent to the largest bistatic range resolution among the three transmitters. Adjacent gridspaces overlap by 50%; that is, the edges of a gridspace's neighboring gridspaces pass through its center (see Figure 2). During an iteration of the PHD filter's time-update step, the number of bistatic range ellipse observations from different transmitters that pass through each gridspace is tallied. Only those gridspaces that have a count of three or more are considered candidates for receiving birth particles. This logic, thus, takes care of localizing the ellipse intersections in range. In other words, the ellipse intersections have been narrowed down to local regions in the (x, y) subset of the target state space.

Now, using only those ellipses that intersect in range, a least-squares solution is implemented to find the possible velocities associated with the Doppler measurements, and the intersections are then narrowed down further using this information (see Section III-D). The result is a set of gridspaces that contain bistatic ellipse observations that intersect in both range and Doppler and, thus, contain possible targets. The birth particles are then placed uniformly with respect to their x and y positions in these gridspaces. To initialize their velocity parameters, the particles are distributed uniformly in a region of

$$[\hat{v} - 2\Delta_{\dot{R}}, \hat{v} + 2\Delta_{\dot{R}}], \tag{4}$$

where $\Delta_{\dot{R}}$ is the largest Doppler resolution (in m/sec) among the transmitters, as specified in Footnote 2, and $\hat{v}$ is the least-squares solution for the velocity observation in the gridspace. The birth particles are divided evenly among the valid target-observation gridspaces. Thus, since the total number of birth particles introduced into the filter is fixed as a configurable parameter, the number of birth particles placed in each "potential-target" gridspace depends on the number of such gridspaces, since each receives an equal share of the total number of birth particles allowed. (Future work could explore allowing the total number of birth particles to vary with the number of "potential-target" gridspaces.) The total weight of all the birth particles sums to the number of birth targets expected at the particular time step. In our simulations, this is assumed to be one. Also, if a gridspace already contains particles that were propagated from the previous time step, then no birth particles are placed in it, since it is assumed that the observations at such a location are caused by an existing target.

Note that the bistatic range resolution remains constant throughout the simulation, so the counting grid can be computed offline before the simulation begins. This makes this grid method practical for near real-time simulation, since computing such a grid at each time step would be impractical.

B. Result of the Range-Resolution Based Grid Technique

Even though the range resolution-based grid method appeared to work reasonably well, it was not robust relative to changes in SNR. This is because the likelihood model that determines the weights in the PHD filter (see Section 5.3 of [1]) is dependent not on the range resolution, but on the range variance, which is itself dependent on SNR.[3] Thus, for example,

[3]The variance of the bistatic range is given as $\sigma_r^2 = \sigma_t^2 \cdot c^2$, where [12] $\sigma_t^2 = \frac{1}{2\beta^2 \mathrm{SNR}(\xi_i)}$, and β is the transmitter bandwidth.

if the SNR were to increase, the range variance then tightens, and the range-resolution based grid no longer positions the particles close enough to the regions of high importance. This is evident, for instance, if the SNR is increased by a factor of 100. The standard deviation of the sensor likelihood function accordingly shrinks by a factor of 10, and the range resolution-based simulation is no longer able to track a target that it had previously tracked. However, if the number of birth particles is also increased by a factor of 100 (needed since we require a factor of 10 increase in both the x and the y dimensions in the target state), the filter is able to track the target again. This demonstrates that the range-resolution grid method does not solve the birth particle placement problem in this application, since target detection, and not just tracking accuracy, still depends on having a large number of particles.

C. The Range-Variance Based Grid Technique

One solution to the problem described in the previous subsection is to space the gridpoints according to range variance. The difficulty with this is that range variance varies with position in the FoV. Thus, the grid will not have evenly spaced gridpoints, as was the case when using range-resolution, but rather it will have gridpoints that are more closely spaced in regions of high SNR (i.e., tight σ_R). Fortunately, this variable grid can be computed offline, since the receiving and transmitting antennas are immobile. The grid in our study was formed by placing most gridpoints at a distance of $2\sigma_R$, and others no closer than $1.7\sigma_R$, from each other (see Figure 2). This flexibility in separation distance was needed to obtain full coverage of the FoV with variably-sized grid spaces, since the algorithm that creates the grid would otherwise mop itself into a corner before it had placed gridspaces throughout the whole FoV. Also, to prevent the grid-making algorithm from being sucked into a black hole around the antennas (where SNR is increasingly large and σ_R increasingly small), our implementation stops placing gridpoints when their gridspace width is less than 10 meters. This limit on gridpoint placement, however, also contributes to the "mopping into a corner" issue of the grid creation algorithm. The resulting grid is shown in Figure 8.

Each gridspace consists of a $4\sigma_R \times 4\sigma_R$ region centered around a gridpoint. Thus, the gridspaces overlap as they did for the range-resolution based grid to obtain adequate coverage. Note that the value of the SNR at the gridpoint is used to determine the $2\sigma_R$ distance to the next gridpoint, and the smallest σ_R among the three possible σ_R values from the three transmitters is used, since the PHD weights are ultimately affected by all three transmitters' σ_R values at any given location. Note the chicken and egg problem evident in using the SNR

at the gridspace's center instead of the largest possible SNR in the gridspace. One can't use the largest possible SNR, because one does not yet know what size the gridspace will be, since it is calculated from the SNR!

Also evident in the grid construction is a tradeoff between whether to use the smallest of the three possible σ_R values to determine the width of each gridspace, or to use the largest of the three. It would be desirable to use the largest of the three values and have large gridspaces, so that no ellipse intersections are missed due to noisy range observations. However, it is also desirable to use the smallest of the three σ_R values, so that targets can be localized adequately and particles distributed in a small area of highest importance, since even if an ellipse intersection is missed due to the observations being too far apart, the intersection would have been given low weight by the PHD filter had it been detected, in any case. Having better target localization is important when there are many false alarms present.

There is also another consideration involved when deciding whether to use the larger gridspace or the smaller one. There will be fewer ellipse observations passing through the smaller gridspace, and so the runtime of the combinatorial Doppler-intersection logic will be shorter. However, because the gridspace is smaller, there will be more gridspaces to process, and so more runs of the Doppler-intersection logic will be required. A detailed analysis of the computational tradeoffs has not been performed, and remains an avenue for future work. Here, we focus on the basic matter of target localization, so we use the smallest σ_R value among the transmitters at each gridpoint for the gridspace.

While constructing the grid offline, the pseudo-inverse matrices needed by the Doppler-intersection logic to compute the least-squares Doppler solutions at each gridpoint are also computed. The Doppler-intersection logic is described in Section III-D.

D. Doppler-Intersection Logic: The Least-Squares Solution

Our current implementation of the grid method identifies all the grid spaces through which more than three ellipses (from different transmitters) pass. This finds the x and y components of the ellipse intersection. To find the $\dot{x}$ and $\dot{y}$ components, every combination of possible triplets of ellipses in the gridspace is formed, and each triplet's associated Doppler observations are used to compute its least-squares solution for target velocity. These velocity estimates are then used to obtain Doppler observation estimates, and the error residual between these estimated Dopplers and the actual observed Dopplers is computed. Only those ellipse triplets whose Doppler error residuals are within a given threshold, which depends

on the Doppler resolutions of the transmitters, are retained as valid ellipse intersections. The least-squares solution algorithm is as follows.

The rate of change in bistatic range that is observed by the radar can be computed from a target's position and velocity:

$$\dot{R} = \frac{(x-x_r)\dot{x}+(y-y_r)\dot{y}}{R_r} + \frac{(x-x_t)\dot{x}+(y-y_t)\dot{y}}{R_t} \tag{5}$$

$$= \left(\frac{(x-x_r)}{R_r} + \frac{(x-x_t)}{R_t}\right)\dot{x} + \left(\frac{(y-y_r)}{R_r} + \frac{(y-y_t)}{R_t}\right)\dot{y} \tag{6}$$

$$= C_t\,\dot{x} + D_t\,\dot{y}, \tag{7}$$

where R_r is the distance between the target and the receiver, and R_t is the distance between the target and transmitter t. Note that C_t and D_t can be computed offline when using a grid system, since x and y are simply the location of the centerpoint of the gridspace.

Thus, with a triplet of PCL-generated Doppler observations, we have an overconstrained system:

$$\begin{pmatrix} C_1 & D_1 \\ C_2 & D_2 \\ C_3 & D_3 \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \dot{R}_1 \\ \dot{R}_2 \\ \dot{R}_3 \end{pmatrix}$$

$$A \qquad v \quad = \quad b, \tag{8}$$

where $\dot{R}_t$ is the Doppler observation from the t-th transmitter, C_t and D_t are computed offline as given above, and $\dot{x}$ and $\dot{y}$ are the target velocity components that we are trying to find. Using the notation in (8), the least squares velocity estimate is

$$\hat{v} = A^T b = (A^T A)^{-1} A^T b. \tag{9}$$

Note that $(A^T A)^{-1} A^T$ is precomputable offline. We now plug $\hat{v}$ back into (8) to obtain the $\dot{R}$ estimate we would receive if the target located in the gridspace actually had this velocity estimate as its true velocity, denoted

$$\hat{b} = A\hat{v}. \tag{10}$$

The error vector residual is computed as

$$e = \hat{b} - b. \tag{11}$$

Our implementation marks an ellipse triplet as a valid intersection if the sum of the squares of the components of its error vector, e, is less than the sum of the squares of twice the Doppler

resolution for each transmitter at the gridspace center. One could imagine experimenting with other criteria.

Note that the Doppler resolution is used here, rather than the Doppler variance. This is because the smallest value of the Doppler variance is limited by the value of the Doppler resolution. That is, the Doppler variance (or, more correctly, standard deviation) can be no less than the Doppler resolution times the wavelength used. Thus, unlike the range variance, which can become much smaller than the range resolution and requires the more complicated variably-spaced grid method to be used, the Doppler observation does not give us the same difficulty. Thus, the simpler Doppler resolution is used.

E. Alternative Intersection-Finding Logic: The Iterative Least-Squares Approach

Another approach to the birth particle placement problem is to use a least-squares iterative technique that solves for both range and Doppler intersections. This method has been implemented by Kees Stolk at NC3A. It does not require an offline computation of a grid, but does require an exhaustive combinatorial search over all the observations at each time step for the ellipse intersections.

Compared to the grid method, the iterative method might require more computation when there are few gridspaces containing three or more ellipse intersections. However, because the grid spaces overlap by half their width, it is possible to have the same ellipses intersect in more than one gridspace. This causes redundant computational work in the grid approach, since the simulation currently searches for all possible ellipse combinations in each gridspace to find the Doppler solutions. Thus, the gridspace technique may at times require more computation than even the exhaustive iterative least-squares approach. A better manner of overlapping the gridspaces, instead of using a simple $2\sigma_R$ spacing, may improve its runtime efficiency.

A problem with the iterative least-squares approach is that it does not always converge to the correct intersection. As illustrated in Figure 3, the algorithm may converge to a ghost target depending on the initial starting point of the algorithm. Additional information, such as angle of arrival observations, would be needed to correctly initialize the algorithm. This problem is not an issue when using the grid approach to find the ellipse intersections, since all valid intersections will be treated as regions of potential targets in the grid method, and the PHD filter handles the task of distinguishing the ghost targets from the real ones.

IV. Improvements to the Multitarget, Multisensor Tracker

This section describes the differences between the multitarget, multisensor PHD-based tracker described in this paper and that presented in [1].

A. Number of Particles

In the simulation presented in [1], the number of birth particles and resampled particles were each specified as fixed parameters. This was changed to the more common and computationally-efficient method used in particle filters, whereby the number of targets present determines the number of particles to be used. In the case of too few targets, a minimum number of particles can also be specified. The number of birth particles at each time step is still roughly fixed at each iteration. The number of particles used in our NC3A scenario is given in Table IV.

B. Sensor Data Collection

We modified our simulation so that only one sensor provides observations at each time step. This was done to match how the real NC3A PCL demonstration system operated at the time of this writing. The simulation waits for all sensor reports to be collected before running the PHD filter. The real receiver collects data over a processing interval on the frequency of one of the transmitters, after which it re-tunes to the frequency of the second transmitter to collect data, and then to the third transmitter, and back to the first, and so on. Note that this requires the observations from the first two transmitters to be propagated in time according to the target motion model, so that they are still sufficiently valid when the PHD filter is run.

C. Antenna Pattern

A more realistic antenna gain pattern was incorporated into the simulation to replace the simple omnidirectional receiving antenna used in [1]. Though the antenna gain pattern varies slightly depending on the frequency of the received signal, the gain pattern for 98 MHz is used in the simulation and is shown in Figure 4. Incorporating the antenna gain pattern affects the signal-to-noise ratio (SNR) and those parameters that depend on SNR, such as range variance and probability of detection. A bird's-eye view of the NC3A transmitter geometry and of the antenna pattern for the area in which all transmitter-receiver pairs have high SNR can be seen in Figure 7(b). The result for each transmitter-receiver pair is shown in Figures 5 and 6. Note the deep nulls introduced at $-90°$ and $90°$ off boresight.

V. RESULTS

This section presents the results of our NC3A simulation, given the improvements and modifications presented in Section IV and using the variably-sized grid method introduced in Section III-C. Tables II and III provide the details on the transmitters and receiver used. The receiver is assumed to be located at NC3A in The Hague, Netherlands. The "x-dist." and "y-dist." are the distances of the transmitters from the receiving antenna in x and y, respectively. The receiver parameters are assumed to be the same as in [1], except that the receiver gain parameter has now been replaced by the antenna gain pattern shown in Figure 4. Additional simulation parameters are given in Table IV. Note the considerable reduction in the number of particles needed from that of [1]. The specified probability of false alarm generates about one to six false alarms per time step.

TABLE II

TRANSMITTING ANTENNA SPECIFICATIONS IN THE NC3A SIMULATION

Location	x-dist.	y-dist.	Frequency (f)	Power (P_T)	Bandwidth (β)
Lopik	49,844 m	-10,127 m	96.8 MHz	100.0 kW	45 kHz
Wieringermeer	117,212 m	91,179 m	97.1 MHz	50.0 kW	45 kHz
Goes	-30,590 m	-65,922 m	99.8 MHz	50.0 kW	45 kHz

TABLE III

RECEIVER SYSTEM SPECIFICATIONS IN THE NC3A SIMULATION

Coherent Processing Interval (CPI)	1.0 sec
Reference Temperature (T_0)	290 K
Noise Figure (NF)	30 dB

The Signal-to-Noise ratios (SNRs) and probabilities of detection (p_D) for each transmitter-receiver pair in this simulation are given in Figures 5 and 6. The region where the p_D is greater than 0.95 for all the transmitters is shown in Figure 7. The centers of the gridspaces of the range-variance based grid that is used to determine the ellipse intersections and to distribute birth particles (see Section III-C) are shown in Figure 8. Both Figure 7(b) and Figure 8 provide good illustrations of the geometry of the scenario configuration and antenna gain

TABLE IV

NC3A SIMULATION PARAMETERS

Simulated time period	1 – 300 sec.
Minimum number of particles	150
Number of particles per target	50
Number of birth particles	100
Probability of false alarm (p_{FA})	10^{-4}
Bistatic radar cross section of targets (σ_{RCS})	10 dB

pattern used in the simulation. The receiver is located in the center of the 160 km $\times$ 160 km field of view (FoV) at location (80 km, 80 km). It is pointed to look at targets over the North Sea, and we assume that its boresight is directed Northwest at an azimuth of $315°$.

Figure 9 contains an example of the PHD filter at time $k = 93$. As indicated, the PHD filter has detected two targets. There are actually four targets present in the simulation at that time; however, two of them are in the null of the receiver antenna pattern and are not detected. Figure 9(c) shows a close-up of the particles surrounding the two detected targets. Each particle is represented as a $(\dot{x}, \dot{y})$-velocity vector located at the particle's (x, y) position.

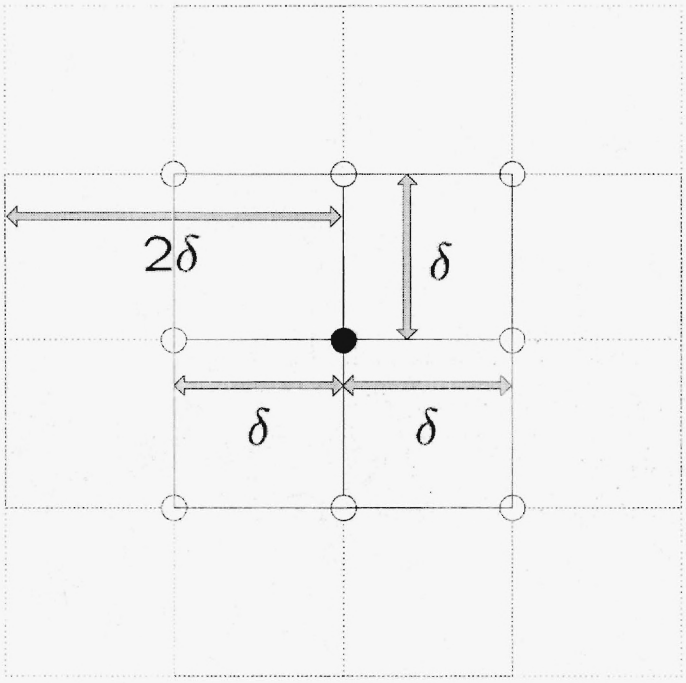


Fig. 2. The gridspacing used in the offline grid. The figure depicts a central gridspace of size $2\delta \times 2\delta$ surrounded by eight adjacent gridspaces. The centerpoint of each gridspace is indicated by a circle. The center gridspace is filled in. In the range-resolution based grid technique of Section III-A, $\delta = \frac{\Delta_{R,max}}{2}$, where $\Delta_{R,max}$ is the largest range resolution among the three transmitters. In the range-variance based grid technique, $\delta \approx 2\sigma_R$, as described in Section III-C.

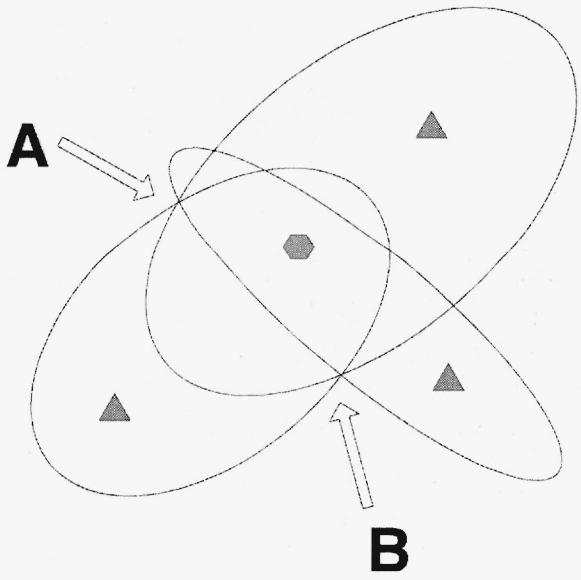


Fig. 3. The iterative least-squares algorithm may converge to either point A or point B, depending on the algorithm's initial starting point. Three bistatic range ellipses are present. The receiver is represented by the hexagon, and the transmitters by the triangles. The configuration shown was created for illustration purposes and is not an exact representation of the simulation scenario.

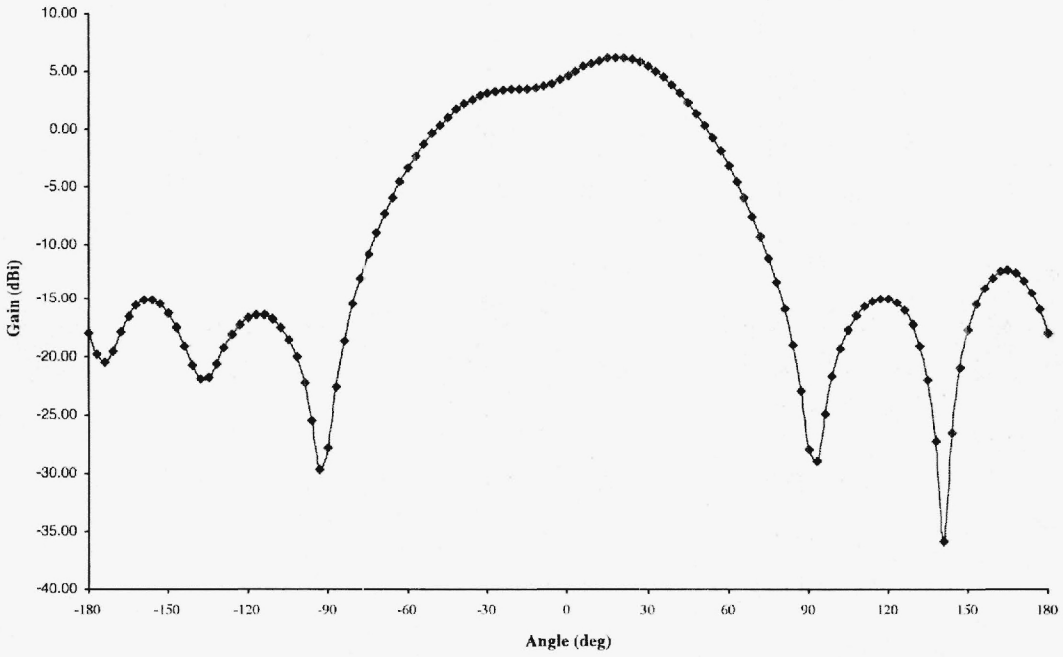
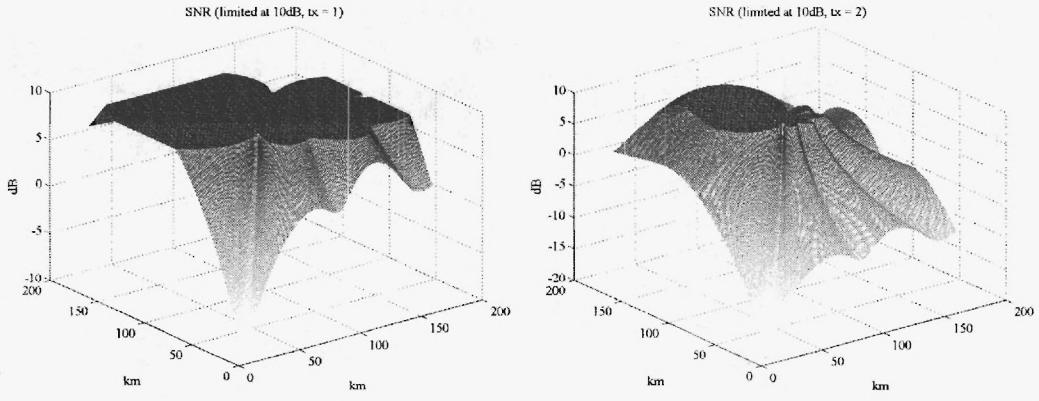
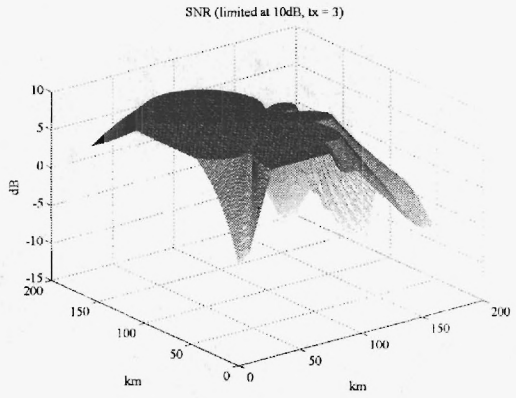


Fig. 4. The assumed antenna gain pattern of the NC3A receiver used in the simulation. The gain pattern is for the E plane of the dipole array, where one dipole is fed and the other terminated. The frequency used is 98 MHz. Plot courtesy of Paul Howland.



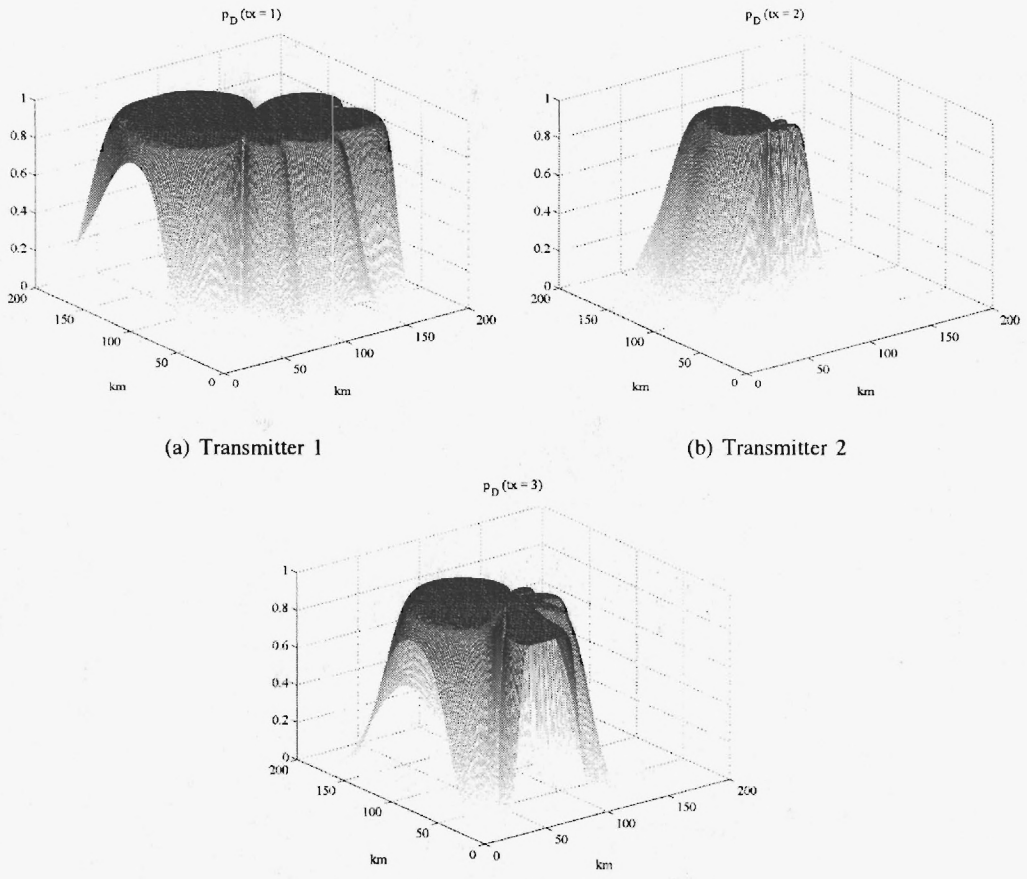
(a) Transmitter 1

(b) Transmitter 2



(c) Transmitter 3

Fig. 5. The signal to noise ratios of each transmitter/receiver pair in the FoV. The SNRs have been truncated at 10dB so that the areas of low SNR may be seen more clearly.

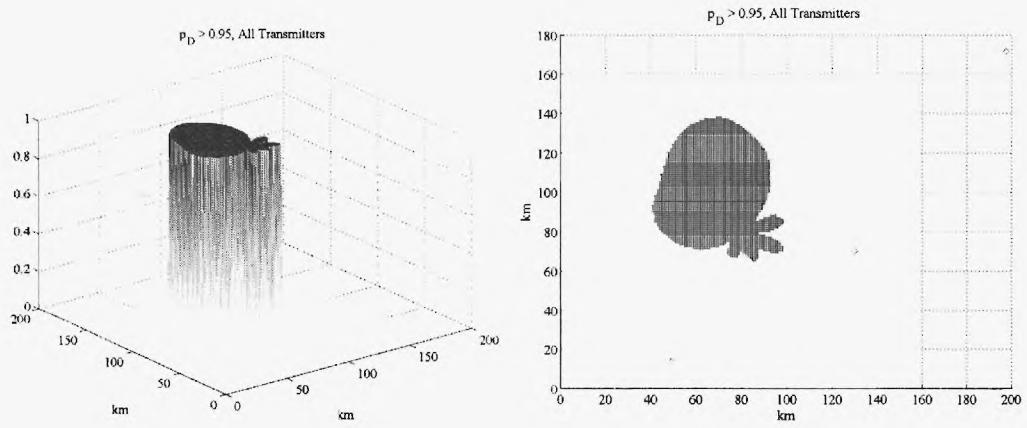


(a) Transmitter 1

(b) Transmitter 2

(c) Transmitter 3

Fig. 6. The probabilities of detection of each transmitter/receiver pair in the FoV.



(a) (b)

Fig. 7. The area in which the probability of detection is greater than 0.95 for all transmitter/receiver pairs. In 7(b), the transmitters are indicated by diamonds, and the receiver is located in the center of the FoV at (80 km, 80 km).

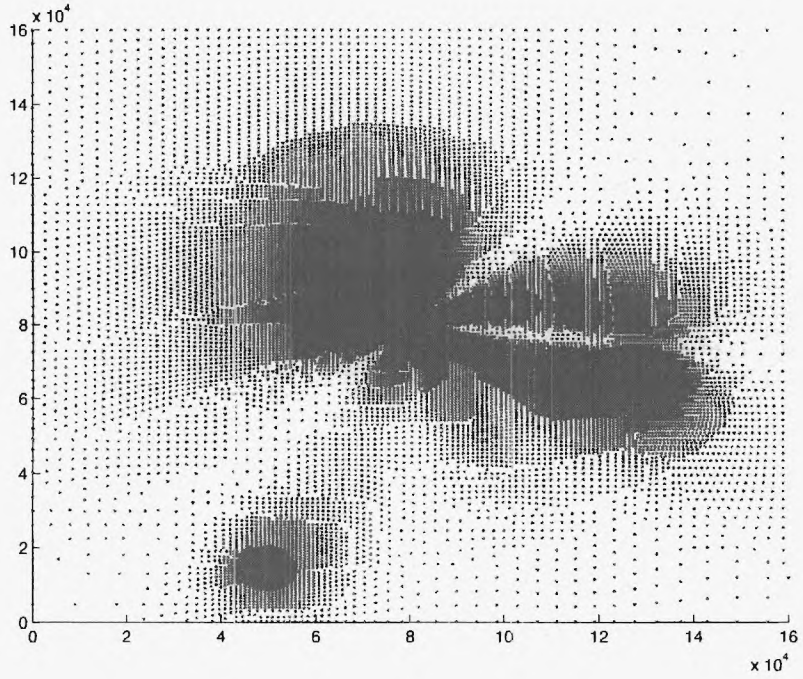


Fig. 8. Gridpoint locations of the bistatic range-variance based grid computed offline. These gridpoints indicate the centers of the gridspaces used to locate ellipse intersections and distribute birth particles. The gridspaces are $4\sigma_R \times 4\sigma_R$ and at least 10 meters wide, and they generally overlap each other by $2\sigma_R$. The gridpoint centers are no closer than $1.7\sigma_R$.

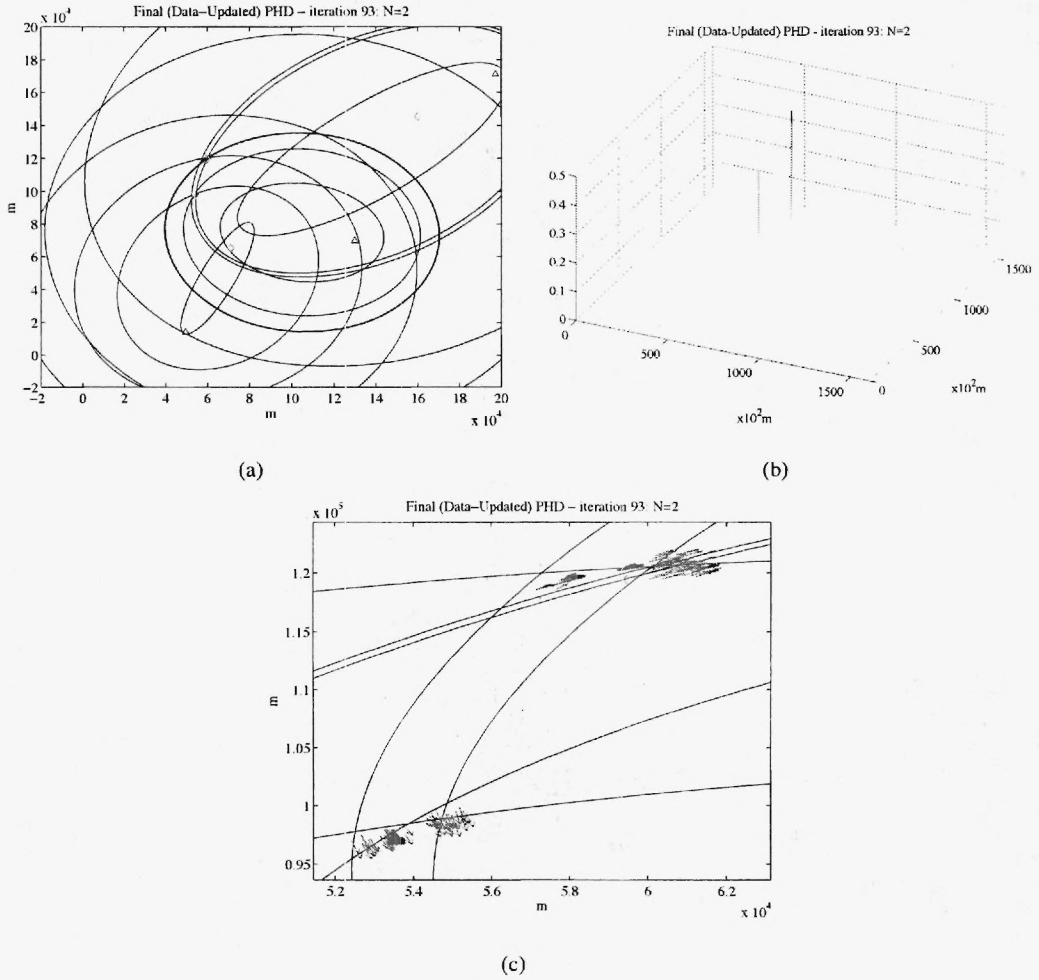


Fig. 9. The PHD particle filter and range ellipses at time $k = 93$ are shown in Fig. 9(a). The receiving antenna is represented by the hexagon, and transmitting antennas by the triangles. The diamonds indicate the actual target positions. Each particle of the filter is pictured. The corresponding particle weights are shown in Fig. 9(b). The sum of the weights is 1.999. A close-up of the particles and two of the targets is given in Fig. 9(c)

For the two detected targets, Table V compares their true states with the estimated states given by the PHD filter.

TABLE V

TARGET STATES AT $k = 93$: ACTUAL VS. ESTIMATED

	Target 1			Target 2		
	TRUE	ESTIMATED	ERROR	TRUE	ESTIMATED	ERROR
x	59,457 m	60,655 m	1198 m	53,774 m	53,399 m	-375 m
y	119,457 m	120,932 m	1475 m	97,268 m	97,914 m	646 m
$\dot{x}$	91.92 m/s	96.95 m/s	5.03 m/s	39.07 m/s	38.50 m/s	-0.57 m/s
$\dot{y}$	91.92 m/s	90.85 m/s	-1.07 m/s	-221.58 m/s	-221.32 m/s	0.26 m/s

A comparison of the estimated target positions versus the true target (x, y) coordinates are given in Figures 10 and 11. Note that there are five targets present. The targets enter at time steps $k = 1, 15, 22, 30$ and 110. However, only two (the ones that enter at $k = 1$ and 22) begin in regions of high enough p_D, so they are the only targets that are immediately detected. The fourth target, which moves off to the West from behind the radar, is detected as it moves into an area of high enough SNR. The total weight at each time step is shown in Figure 12. One can see that at $k = 24$, the second target is detected. (Recall that the previous iteration of the PHD filter is at $k = 21$, since we wait until all three transmitters have collected data before running the PHD filter.) The estimated target velocity components are contrasted with the true target velocities in Figures 13 and 14. The errors in position and velocity estimates for the 300 time steps are plotted in Figure 15. Given that the range resolution in the simulation is around 6.7 km, and the Doppler resolution is a little over 3 m/sec, the resulting position and velocity errors are quite reasonable.

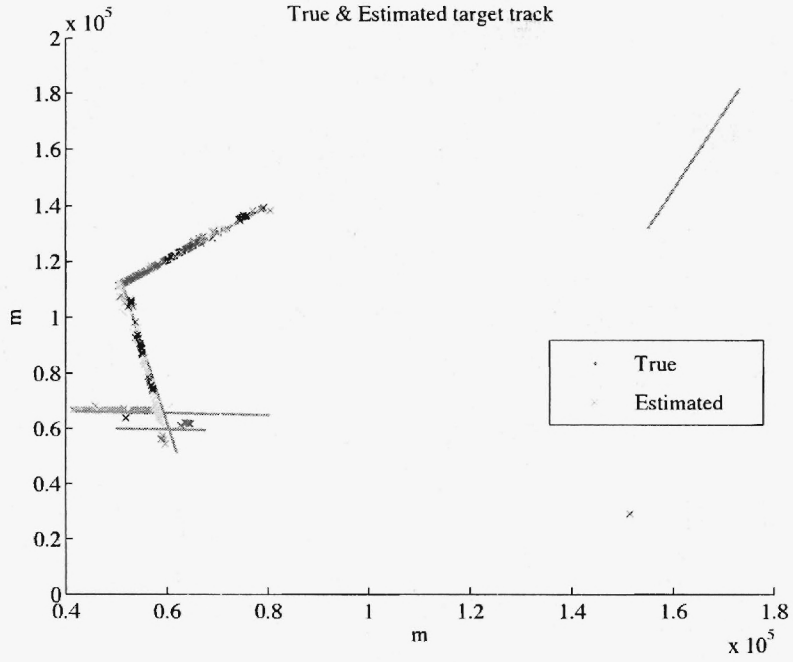


Fig. 10. Actual vs. estimated target locations over first 300 iterations.

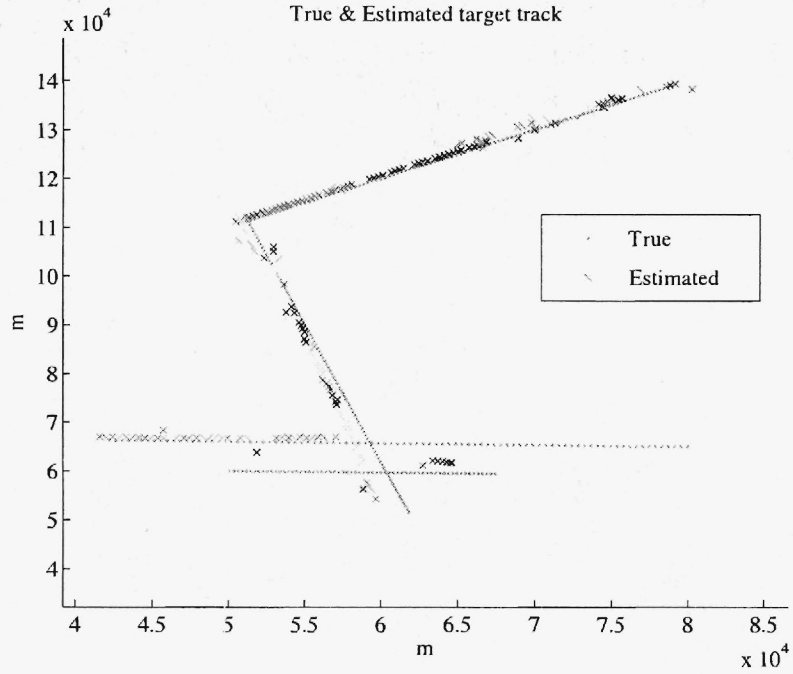


Fig. 11. Close-up of actual vs. estimated target locations over first 300 iterations.

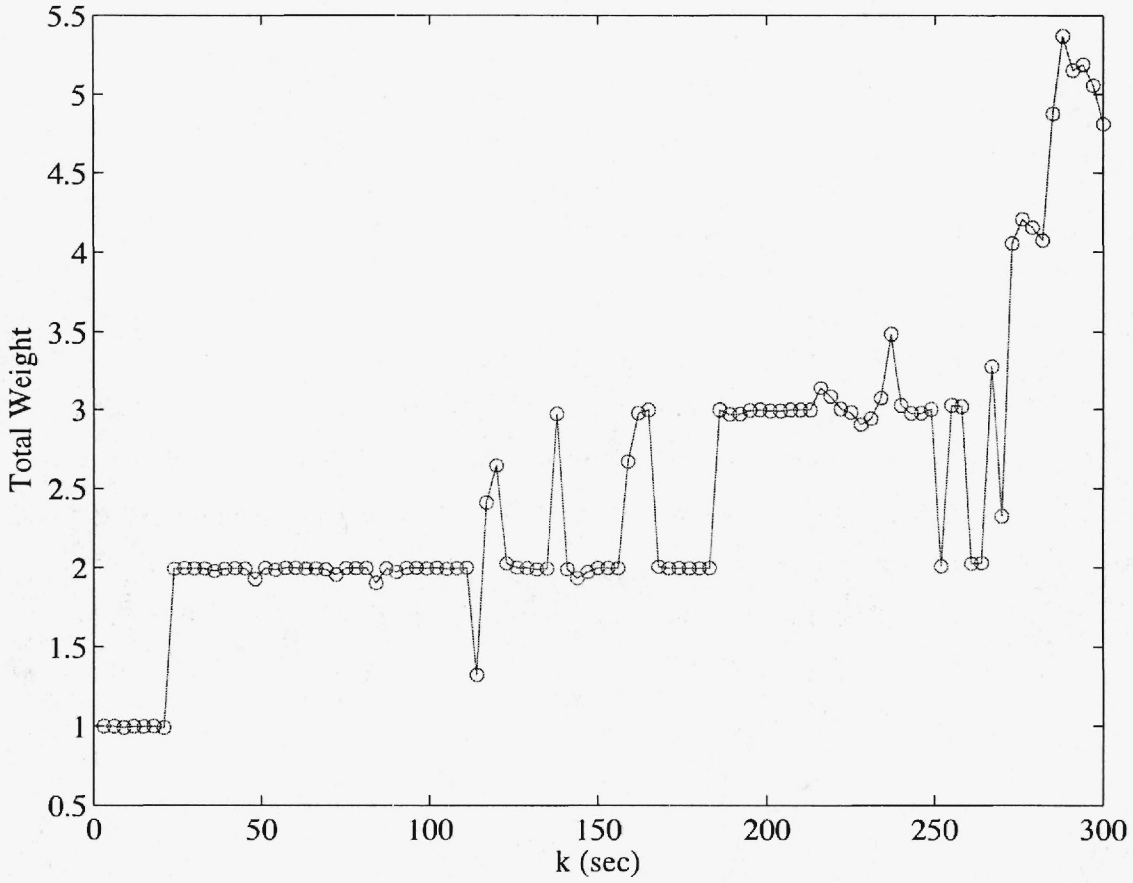


Fig. 12. Total weight at each time step. The expected number of targets at each time step is found by rounding the total weight to the nearest integer.

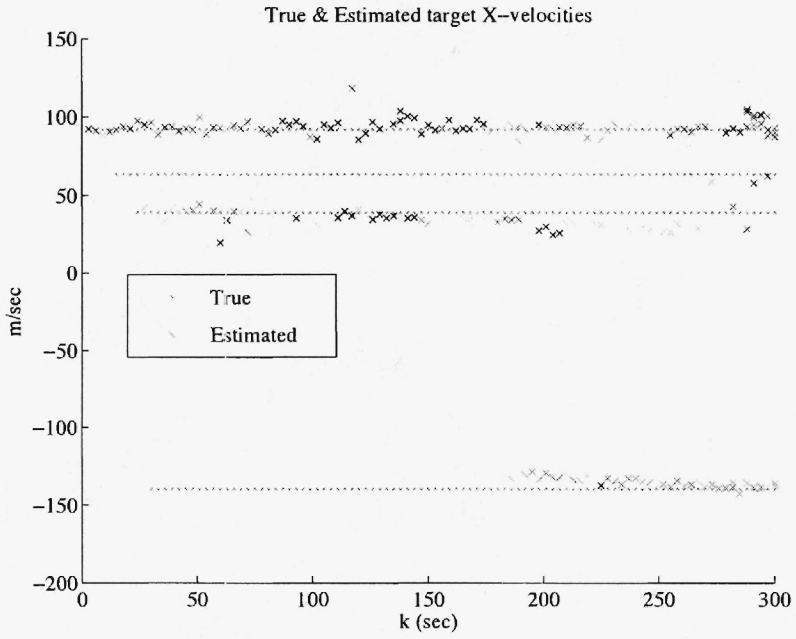


Fig. 13. Actual vs. estimated target X-velocities over first 300 iterations.

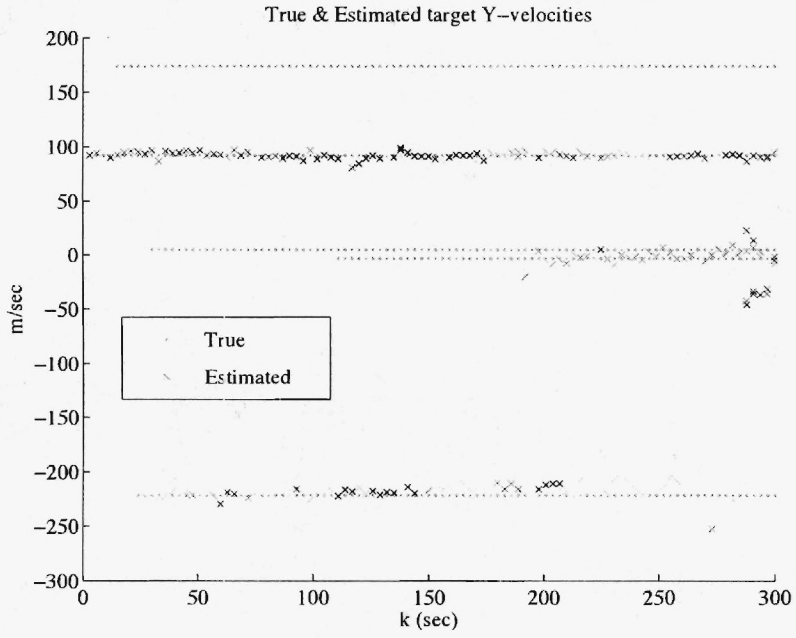


Fig. 14. Actual vs. estimated target Y-velocities over first 300 iterations.

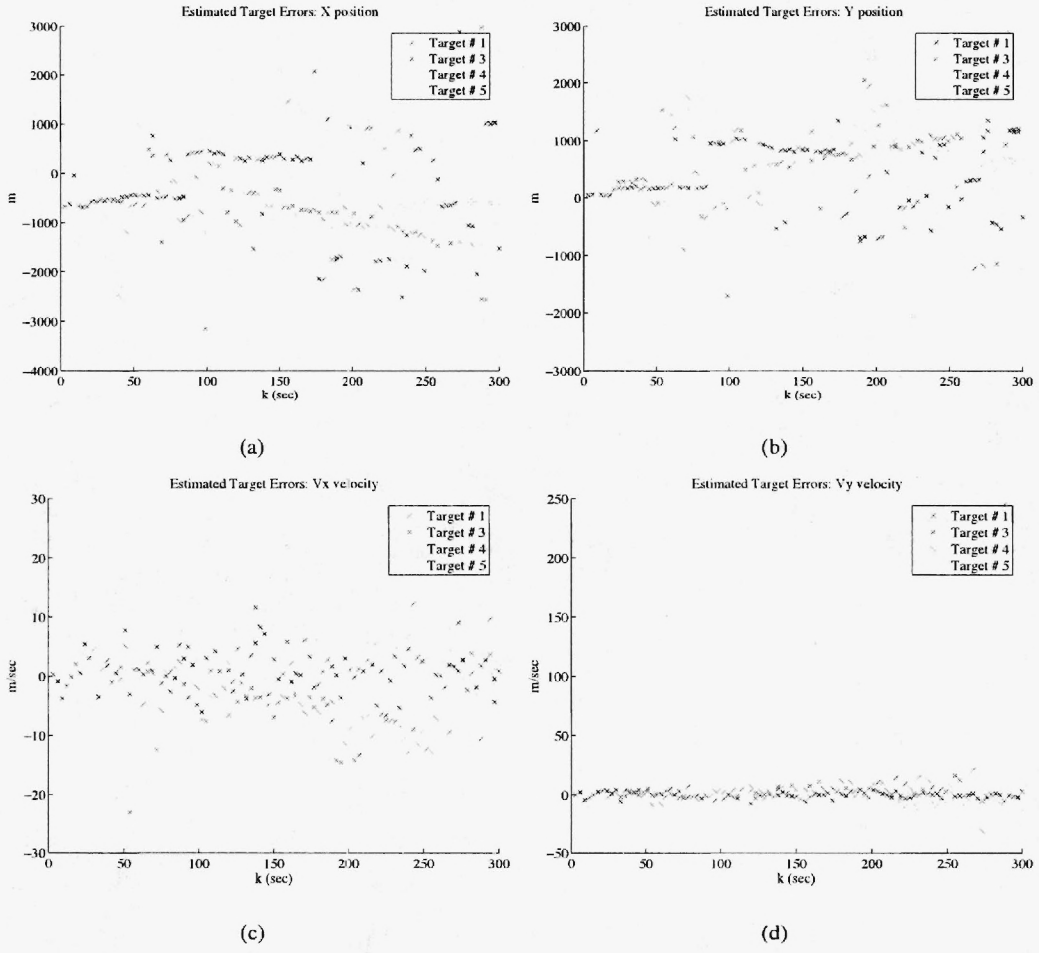


Fig. 15. Errors in target state estimation by PHD filter. The range resolution is around 6.7 km, and the Doppler resolution is a little over 3 m/sec.

VI. Conclusions

When the pre-computed σ_R-spaced grid is used, tracking functionality no longer appears to be dependent on SNR. Furthermore, not only does one not need to increase the number of particles to maintain tracking ability, as was the case when the range-resolution spaced grid was used, but one can even track targets with fewer particles altogether. In our simulation, the number of particles was reduced by a factor of 10, and this caused no adverse effects in tracking performance.

Though it was suggested in [1] that logic be added to restrict particles to areas of high SNR, this logic was found to be unnecessary in light of our new birth-particle placement method. The placement restriction was originally suggested to prevent target number overestimation when birth particles were spontaneously placed in areas of low SNR. In such a case, the filter had to assume that there were targets present in these areas of low p_D, since it had no observable data to contradict the assumption. This was the case when birth particles were simply spread around the edges of the FoV, as done in [1]. In the new birth-particle placement method, particles are placed at observation intersections. Hence, the filter has adequate observability in those regions and can correctly handle particle survivability. In Fig. 12, the overestimation indicated in the last few time steps can be attributed to an interplay between false alarms and the ambiguity in triangulating targets for which not all sensors have high p_D.

The major bottleneck in processing performance is, by far, the ellipse-intersection logic. Even if the iterative least-squares technique is used, as described in Section III-E, instead of the grid techniques, an exhaustive combinatorial search is still required to find all of the ellipse intersections. This is computationally feasible when there are few false alarms (on the order of 20-30 per time step), but it becomes quite intractable when the number of false alarms is much greater (such as when $p_{FA} = 10^{-2}$, as described in [1]).

Thus, an even smarter birth particle placement method may be desired to allow the filter to run in real-time in the presence of many false alarms. One possible remedy would be to fix the probability of false alarm at a reasonable level and simply settle for a lower probability of target detection. If parallel hardware is available, it may also be possible to achieve an immediate speed up in runtime by parallelizing the combinatorial ellipse intersection logic.

In the future, we hope to test the PHD filter on real data that will be collected from the passive radar at NC3A.

ACKNOWLEDGMENTS

This work was enabled via startup funds from the School of Electrical Engineering at the Georgia Institute of Technology, the Demetrius T. Paris Junior Professorship, U.S. Air Force Office of Scientific Research grant F49620-03-1-0340 and the support of Dr. Paul Howland and Dr. Rene van der Heiden of NATO C3 Agency. We thank Dr. Ronald Mahler of Lockheed Martin MS2 Tactical Systems for all the help he has provided us in our work with the Probability Hypothesis Density.

Thanks to Dr. Paul Howland, Martin Tobias was fortunate to have the opportunity to intern at the NATO Consultation, Command and Control Agency (NC3A) in The Hague, Netherlands, during the summer of 2005.

Aaron Lanterman would like to express gratitude to members of the passive radar group at Lockheed Martin Mission Systems, with whom he had the pleasure of interacting while a postdoctoral researcher at the University of Illinois at Urbana-Champaign.

REFERENCES

[1] M. Tobias and A. D. Lanterman, "Probability hypothesis density-based multitarget tracking with bistatic range and Doppler observations," *IEE Proc. Radar, Sonar and Navigation*, vol. 152, pp. 195–205, June 2005.

[2] T. Zajic and R. Mahler, "A particle-systems implementation of the PHD multitarget tracking filter," in *Signal Processing, Sensor Fusion, and Target Recognition XII*, I. Kadar, Ed., vol. Proc. SPIE 5096, 2003, pp. 291–299.

[3] T. Zajic, B. Ravichandra, R. Mahler, R. Mehra, and M. Noviskey, "Joint tracking and identification with robustness against unmodeled targets," in *Signal Processing, Sensor Fusion, and Target Recognition XII*, I. Kadar, Ed., vol. Proc. SPIE 5096, 2003, pp. 279–290.

[4] B.-N. Vo, S. Singh, and A. Doucet, "Sequential Monte Carlo implementation of the PHD filter for multi-target tracking," in *Proc. FUSION 2003*, Cairns, Australia, 2003, pp. 792–799.

[5] ——, "Sequential Monte Carlo methods for multi-target filtering with random finite sets," *IEEE Trans. Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1224–1245, October 2005.

[6] M. Tobias and A. D. Lanterman, "A probability hypothesis density-based multitarget tracker using multiple bistatic range and velocity measurements," in *Proc. of the 36th Southeastern Symposium on System Theory*, Atlanta, GA, March 2004, pp. 205–209.

[7] ——, "Multitarget tracking using multiple bistatic range measurements with probability hypothesis densities," in *Signal Processing, Sensor Fusion, and Target Recognition XIII*, I. Kadar, Ed., vol. Proc. SPIE 5429, April 2004, pp. 296–305.

[8] H. Sidenbladh, "Multi-target particle filtering for the probability hypothesis density," in *Proc. FUSION 2003*, Cairns, Australia, 2003, pp. 800–806.

[9] D. Clark, J. Bell, Y. de Saint-Pern, and Y. Petillot, "PHD filter multi-target tracking in 3D sonar," in *Oceans 2005 - Europe*, vol. 1, Brest, France, 20-23 June 2005, pp. 265–270.

[10] D. Clark and J. Bell, "Bayesian multiple target tracking in forward scan sonar images using the PHD filter," *IEE Proc. Radar, Sonar and Navigation*, vol. 152, no. 5, pp. 327–334, October 2005.

[11] R. Bose, A. Freedman, and B. D. Steinberg, "Sequence clean: A modified deconvolution technique for microwave images of contiguous targets," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 38, no. 1, pp. 89–97, Jan. 2002.

[12] D. Barton, *Modern Radar System Analysis*. Artech House, Inc., 1988.

Appendix H

M. Tobias and A.D. Lanterman, "Multipath Effects and the PHD Particle Filter," *IEE Proc. Radar, Sonar, and Navigation*, to be submitted.

Multipath Effects and the PHD Particle Filter

Martin Tobias† and Aaron D. Lanterman
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250 U.S.A.
E-mail: mtobias@ll.mit.edu, lanterma@ece.gatech.edu

Abstract

The Probability Hypothesis Density (PHD) particle filter appears to be a promising tool for multitarget tracking in passive radar systems that exploit "illuminators of opportunity." This paper explores multipath effects in order to stress the robustness of the PHD filter to varying signal-to-noise ratios (SNR) and probabilities of detection, particularly in cases where the true probabilities of detection differ from those assumed by the filter. We find that the behavior of the PHD filter is highly sensitive to the changes in the modeling of SNR.

I. Introduction

Since its introduction by Mahler [1], the target tracking community has evidenced gradually increasing interest in the Probability Hypothesis Density (PHD) approach to target tracking. In Bayesian multiple-target tracking applications, the integral of the PHD over a given area gives the expected (in the Bayesian estimation sense) number of targets in that area. Just as particle filters have become popular for propagating Bayesian posterior densities in single-target tracking, so have they become popular for propagating Bayesian PHDs in multiple-target tracking [2], [3]. The PHD approach conveniently fuses data from multiple sensors, since it avoids the need for explicit target-to-track associations at the fusion stage or for explicit logic to determine the number of targets. Targets are located by extracting peaks from the PHD.

† Currently at M.I.T. Lincoln Laboratory.

Some initial simulations [4]–[6] suggested that the PHD particle filter showed promise in passive radar applications, in which multiple targets are tracked using reflections of pre-existing signals such as FM radio broadcasts. Two main advances presented in [6] over earlier work presented in [4] were: 1) a new peak extraction algorithm that was more effective and computationally efficient than the expectation-maximization algorithm used in [4] and 2) a computationally efficient method of placing "birth particles," which represent possible target states in the PHD particle filter framework, by searching for intersections of range ellipses at predetermined points on a *variably-spaced* grid, where the grid spacing is based on the variances of the range measurements.

However, these initial simulations were idealized in that they did not include any multipath effects that one would expect to occur in a real system. Multipath effects, which are detailed in the remaining sections of this introduction, result in signal-to-noise ratios (and hence probabilities of detection) that may vary drastically from one point in space to the next. In practice, these multipath effects are difficult to predict precisely because of variations in terrain and atmospheric conditions, so it may not always be feasible to specify their exact effects in advance.

To our knowledge, this paper presents the first study of the performance of a PHD filter under such rapid and perhaps imperfectly modeled SNR variations. To avoid excessive repetition with previous works, we refer the reader to [6] for the theory behind the PHD filter, as applied in this passive radar context, as well as for the most relevant implementation details. Hence, Section II just focuses on the addition of multipath, and some conclusions and directions for future work are presented in Section III.

A. *Multipath Effects*

Multipath effects arise because of the interaction of the radio waves with the ground and other physical surfaces as they propagate from the transmitting antenna to the receiver. These effects are manifest in the signal-to-noise (SNR) ratio via the squared propagation factors, F_R^2 and F_T^2, in the equation for the bistatic radar constant K used to determine the SNR:

$$\mathrm{SNR}(\xi_i) = \frac{K}{R_T^2 R_R^2}, \tag{1}$$

where R_T and R_R are the distances between the target's location and the sensor's transmitting and receiving antennas, respectively, and

$$K = \frac{P_T\, G_T\, G_R\, \lambda_j^2\, \sigma_{rcs}\, F_T^2 F_R^2}{(4\pi)^3 k\, T_0 (\frac{1}{CPI})(NF)}, \tag{2}$$

where P_T is the transmitted power, G_T and G_R are the gains of the transmit and receive antennas, λ_f is the wavelength of the transmitted wave, σ_{rcs} is the bistatic RCS of the target, k is Boltzmann's constant, T_0 is the system temperature, CPI is the coherent processing interval. In passive radar studies, the noise figure NF typically encapsulates the effects of out-of-band interference in addition to receiver noise. In the bistatic passive radar case, the F_T^2 term accounts for the multipath effects in the transmitter to target path, while the F_R^2 term accounts for the effects in the target to receiver path.

In the multipath simulations presented in this paper, the propagation factors are computed using the physical characteristics of the NC3A passive radar described in [7]. A three-dimensional flat-earth model is used, and targets are assumed to fly at an altitude of 7315 m (approx. 24,000 ft) above ground level. A vertically-polarized receiver is located at a height of 20 m, and the transmitters are all assumed to be located at a height of 375 m above ground. A depiction of this scenario appears in Figure 1. This three-dimensional, flat-earth model is incorporated into the simulations, so that the bistatic range and Doppler observations take the target heights into account. However, the PHD-based tracker presented here does not currently take target altitude into account (as is the case with many two-dimensional tracking algorithms). That is, no z or $\dot{z}$ parameters are added to the particle states. This results in a slight bias in the observed target locations relative to the true target locations, as evident in Figure 8. Because the goal of the research is to present an initial evaluation of the performance of the PHD filter, we begrudgingly leave in this bias, since it does not affect our evaluation of the PHD and adding z-states would require a considerable amount of additional computation. Indeed, the bistatic range ellipses would become ellipsoids, and the offline grid and iterative least-squares techniques for placing birth particles described in [5], [6] would need to be expanded to three dimensions.[1]

B. Propagation Factors

Since the squared propagation factors F_T^2 and F_R^2 have similar forms, a generic term F^2 will be used to represent both in the following discussion. Using the equations and plots

[1]Furthermore, it is difficult to obtain good altitude information on low-flying targets because of increased geometric dilution of precision (GDOP), which is a measure of the loss of a radar system's ability to locate targets because of poor geometries. Many surveillance radars are used to track targets in only two dimensions.

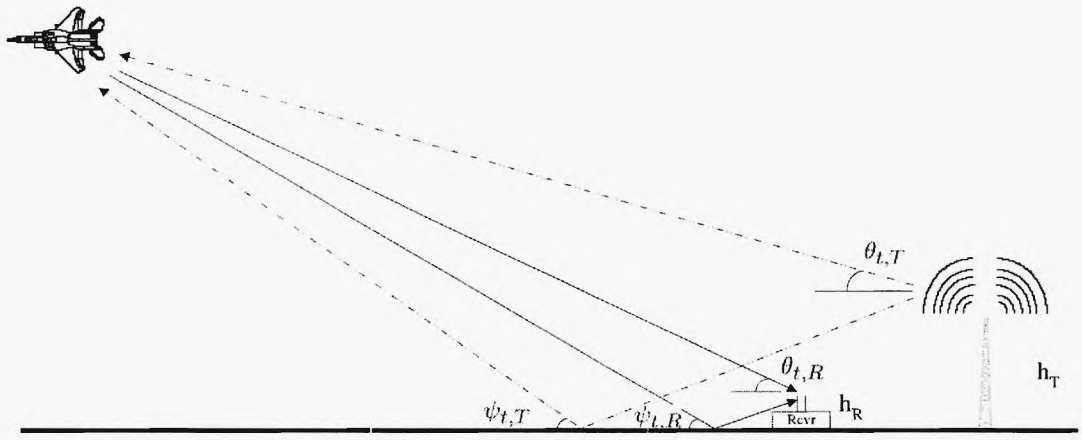


Fig. 1. A diagram of the multipath between a transmitter and the receiver. The dashed lines are the paths of the radio signal from the transmitting antenna to the target and determine F_T^2, while the solid lines are the paths of the echo signal from the target to the receiver that determine F_R^2. The elevation angles of the target relative to the transmitter and receiver are indicated by $\theta_{t,T}$ and $\theta_{t,R}$, respectively. The grazing angles are $\psi_{t,T}$ and $\psi_{t,R}$, and the heights of the antennas are h_T and h_R for the transmitter and receiver, respectively.

given by Barton in Section 6.2 of [8], the squared propagation factor of a path is

$$F^2 = 1 + \rho^2 + 2\rho\cos\left(\phi + 2\pi\frac{\theta_t}{\theta_n}\right), \tag{3}$$

where $\theta_n = \frac{\lambda}{2h}$, λ is the wavelength of the propagating wave, and θ_t is the elevation angle of the target relative to the antenna, and h is either the height h_R of the receiver, in the case of F_R^2, or the height h_T of the transmitter, in the case of F_T^2. This assumes that $h_T >> h_R$. The reflection phase shift term, ϕ, is set to zero in the simulation.[2] The surface reflection coefficient, ρ, is defined to be $\rho = \rho_0\rho_s\rho_v$, where ρ_0 is the magnitude of the complex Fresnel reflection coefficient of the surface, ρ_s is the specular scattering coefficient of a rough surface, and ρ_v is the vegetative absorption coefficient. The Fresnel reflection coefficient depends on the grazing angle ψ and the complex dielectric constant ϵ_c of the surface material. For vertical polarization, its magnitude is given by

$$\rho_0 = \left|\frac{\epsilon_c\sin\psi - \sqrt{\epsilon_c - \cos^2\psi}}{\epsilon_c\sin\psi + \sqrt{\epsilon_c - \cos^2\psi}}\right|, \tag{4}$$

where $\epsilon_c = \epsilon_r - j60\lambda\sigma_e$, and ϵ_r and σ_e are respectively the relative dielectric constant and conductivity of the surface. Since the NC3A radar we wish to model looks out over the sea,

[2]Given the wavelengths and grazing angles present in the simulation, ϕ should in fact be set to 180° (see pgs. 292-293 of [8]), which would result in a simple phase shift of the cyclic propagation factor F. Such a phase shift would not be expected to effect the qualitative nature of the PHD filter tracking.

we assume that the reflection surface is that of salt water, and hence set $\epsilon_r = 75$ and $\sigma_e = 5$ mho/m [8]. We also assume that the sea water is not a rough surface and that $\rho_s = 1$. We assume also that there is no vegetation with a thickness greater than a wavelength present everywhere, and so we assume that $\rho_v = 1$.

Thus, as noted in [8], the propagation factor F varies cyclically in target elevation angle with a period of θ_n between a maximum of $F_{max} = 1 + \rho$ and a minimum of $F_{min} = 1 - \rho$.

Figure 2 shows the magnitudes of F_R^2 and F_T^2 for each of the transmitters used in our NC3A simulation. Note that F_R^2 depends on the transmitter only to determine which λ to use in computing F_R^2. Thus, F_R^2 does not vary much based on which transmitter is used. In contrast, the plots of F_T^2 for each transmitter do vary significantly. Since the height of the receiver is less than the height of the transmitters, the rate of the cyclic variation of F_R^2 is slower than that of F_T^2, as evidenced by the rings in Figure 2. Figure 3 contains the two-way power ratio, $F_T^2 F_R^2$, used to calculate the SNR, via (1), for each of the transmitters involved in the NC3A simulation; one can see the rings arising from both F_R and F_T. Table I gives the minimum and maximum values of the magnitude of the two-way power ratio.

TABLE I

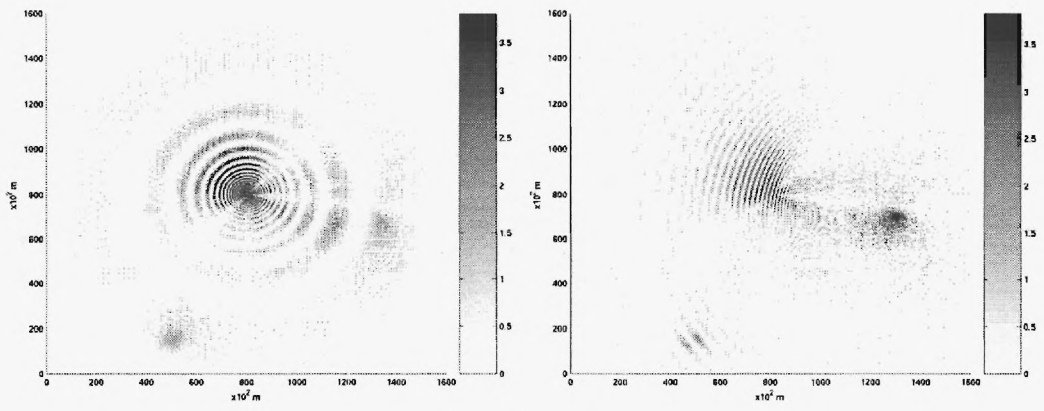
MINIMUM AND MAXIMUM MAGNITUDES OF TWO-WAY, MULTIPATH POWER RATIO

	Minimum $\lvert F_T^2 F_R^2 \rvert$	Maximum $\lvert F_T^2 F_R^2 \rvert$
Transmitter 1	0.0006	11.35
Transmitter 2	0.0031	6.58
Transmitter 3	0.0011	9.93

II. SIMULATION RESULTS

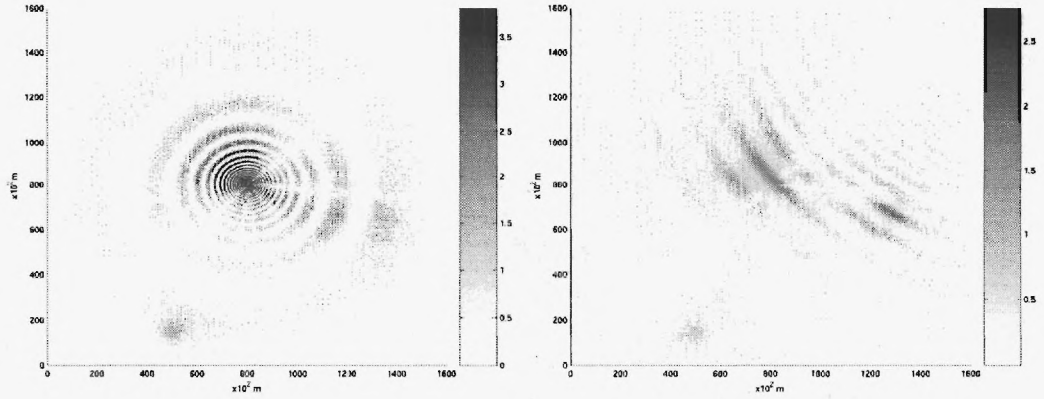
A. *Multipath Effects*

Having incorporated the multipath model into our simulated observations, we tested the PHD filter for its robustness to slowly fluctuating reflections from the target and the resulting changes in the probability of detection (p_D). We want to determine if the PHD filter can still track targets using range and Doppler observations, as it did in [5], [6], by placing birth particles at the intersections of the bistatic range ellipses and Doppler observations using an offline-computed range-variance based grid that was generated without taking F_R and F_T into account, even though the observations now incorporate the multipath effects.



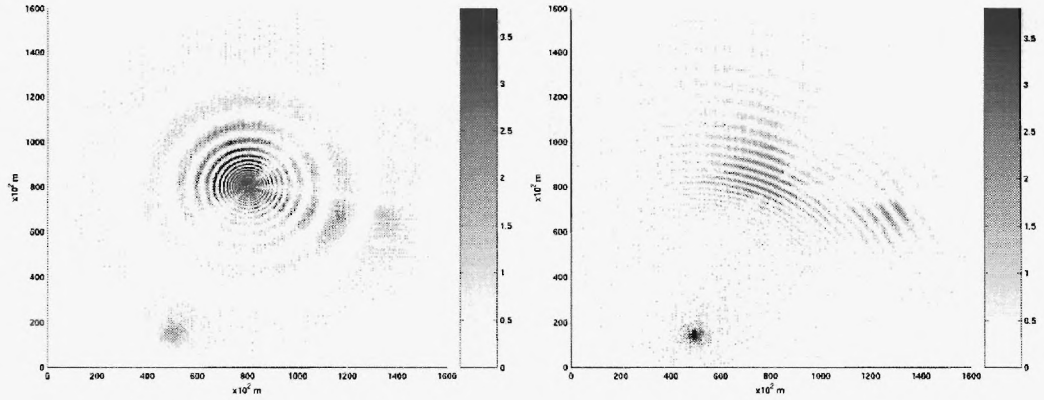
(a) Transmitter 1, F_R^2

(b) Transmitter 1, F_T^2



(c) Transmitter 2, F_R^2

(d) Transmitter 2, F_T^2



(e) Transmitter 3, F_R^2

(f) Transmitter 3, F_T^2

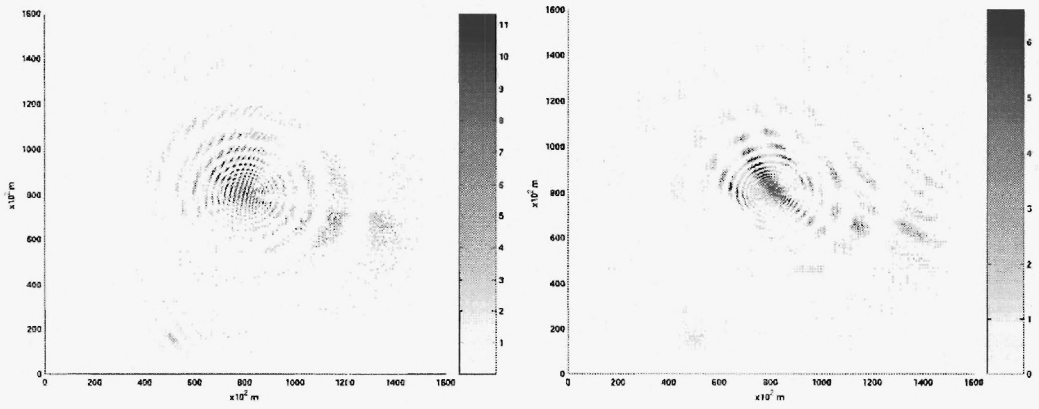
Fig. 2. The magnitudes of the squared propagation factors, F_R^2 and F_T^2, used in the NC3A simulation, for a target located at an altitude of 7315 m. The values plotted are those at the centerpoints of the gridspaces in the range-variance based grid.

The simulations in this section consist of two targets. Target 1 is present at time step $k = 1$, where each discrete time step represents one second, and moves at a speed of 130 m/sec at a heading of $45°$ in azimuth. Target 2 enters at time step $k = 22$ and travels at 225 m/sec with an azimuthal heading of $170°$. The values of the SNR and p_D for the two targets throughout the simulation are displayed in Figures 4 and 5.

The performance of the multitarget tracker in estimating the number of targets is shown in Figure 6. The PHD filter, as implemented, exhibits frequent overestimation, and occasional underestimation, of the number of targets. Figures 7 and 8 compare the real target locations to those found by the PHD filter and peak-extraction algorithm described in [5], [6], and Figure 9 compares the velocities found to the true target velocities. The errors in the estimated target positions and velocities are displayed in Figure 10. These errors are slightly greater than those found when multipath was not included in the simulation. This is somewhat due to the bias introduced by the three-dimensional flat-earth model, as described in Section I, as well as to the incorrect tracking of Target 2 in the later stages of the simulation. Note how the estimated location of Target 2 starts to veer off of the true path of the target before it enters an area of low SNR for all its transmitter-target-receiver links.

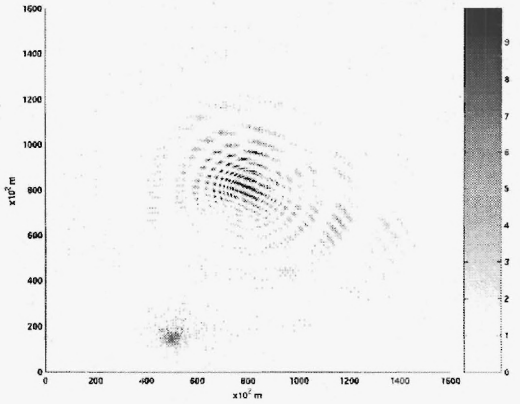
As seen in the figures, both targets are dropped by the range and velocity tracker upon entering regions of low SNR and then were detected again upon re-entering regions of high SNR. Our explanation for this behavior is that, in the regions where the targets are dropped, the PHD filter assumes that the targets are in areas of high SNR. Yet, because of multipath effects, the targets are actually in regions of low SNR. Thus, the PHD filter ends up missing observations in an area of assumed high p_D; hence, the targets are dropped. This is the issue with which Erdinc et al. [9] are concerned. They claim that the PHD filter drops targets abnormally quickly, relative to a Markov chain model, due to the $(1 - p_D)$ term in the Date Update equation. This issue requires further study by the PHD-based target-tracking community, in general.

Note that a ghost target, here defined as a point at an intersection of range ellipses that does not correspond to a true target, is detected and followed when the PHD filter loses track of Target 1.



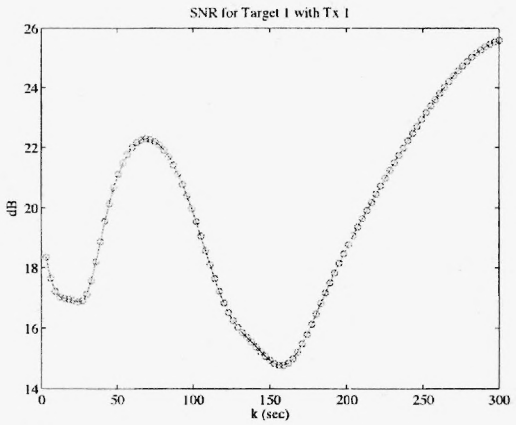
(a) Transmitter 1, $F_R^2 F_T^2$

(b) Transmitter 2, $F_R^2 F_T^2$

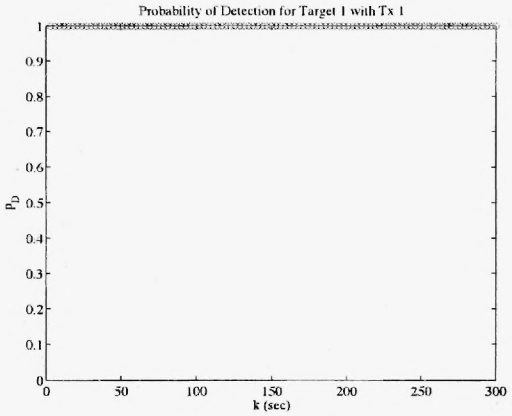


(c) Transmitter 3, $F_R^2 F_T^2$

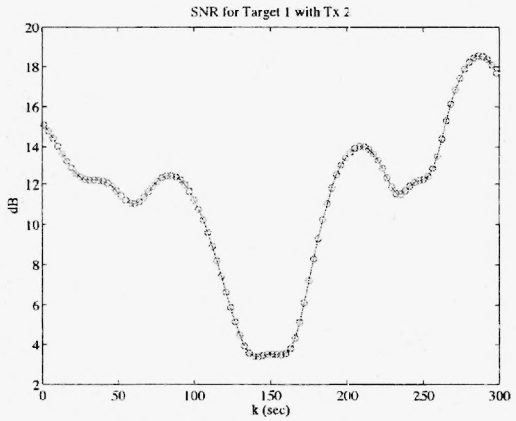
Fig. 3. The magnitudes of the two-way power ratio, $F_R^2 F_T^2$, for each of the transmitters used in the NC3A simulation, for a target located at an altitude of 7315 m. The values plotted are those at the centerpoints of the gridspaces in the range-variance based grid.



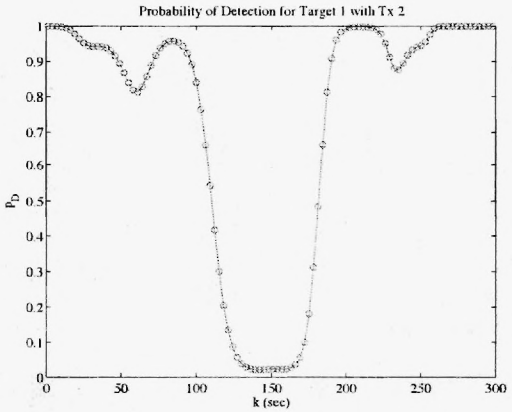
(a) Transmitter 1, SNR



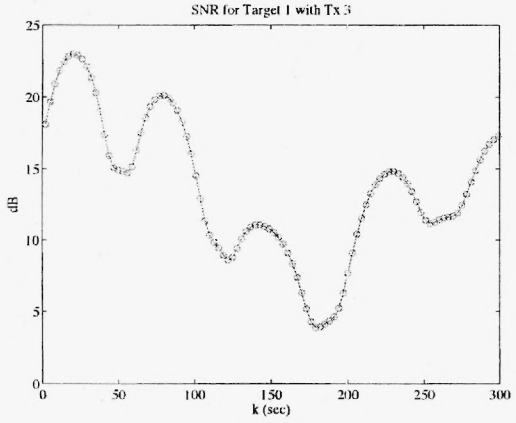
(b) Transmitter 1, p_D



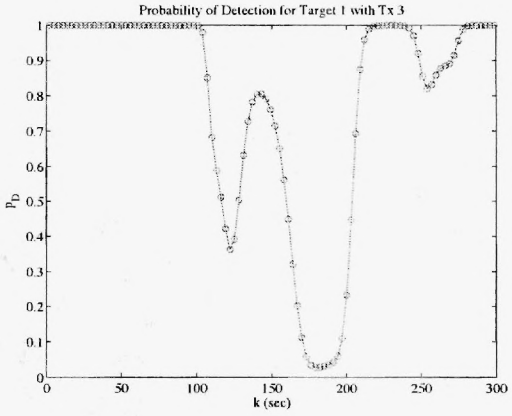
(c) Transmitter 2, SNR



(d) Transmitter 2, p_D

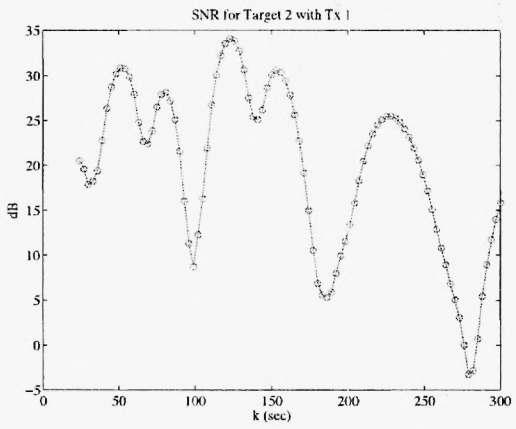


(e) Transmitter 3, SNR

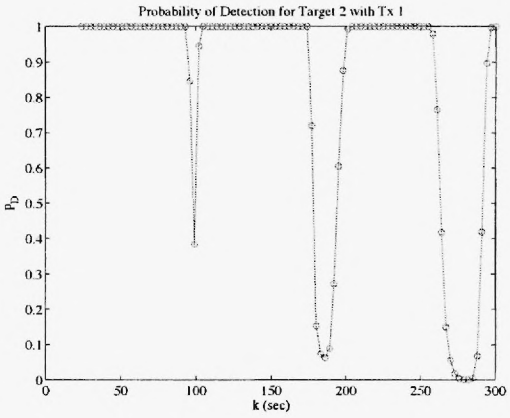


(f) Transmitter 3, p_D

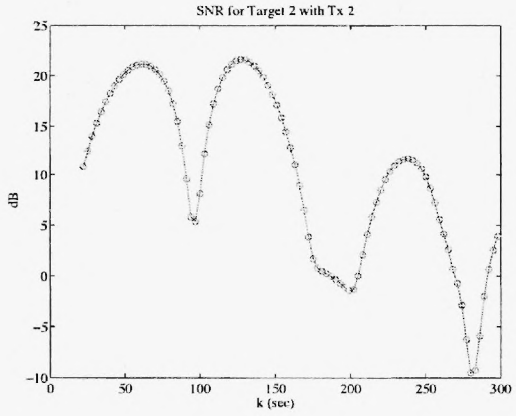
Fig. 4. The signal to noise ratios and probabilities of detection of Target 1 for each transmitter with multipath effects present. The circles indicate the actual values sampled.



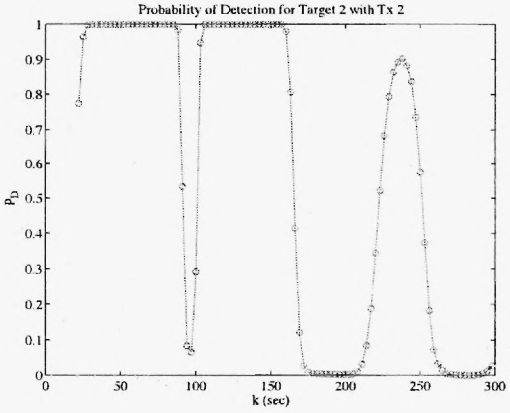
(a) Transmitter 1, SNR



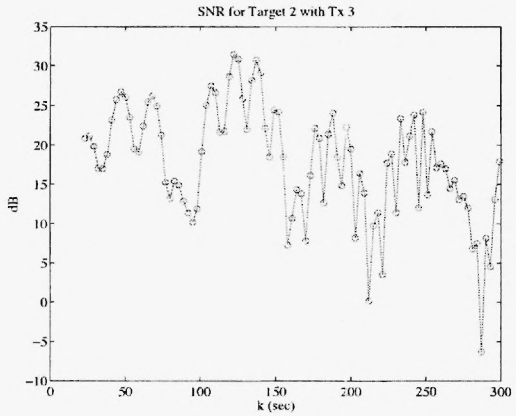
(b) Transmitter 1, p_D



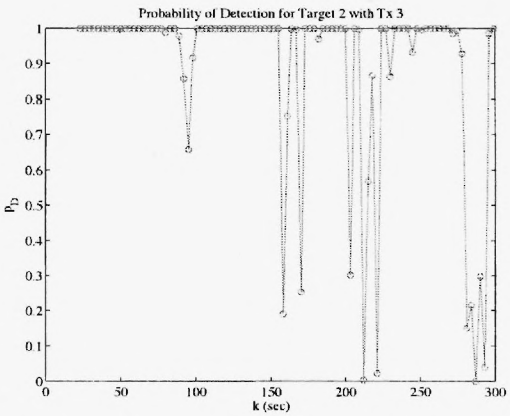
(c) Transmitter 2, SNR



(d) Transmitter 2, p_D



(e) Transmitter 3, SNR



(f) Transmitter 3, p_D

Fig. 5. The signal to noise ratios and probabilities of detection of Target 2 for each transmitter with multipath effects present. The circles indicate the actual values sampled.

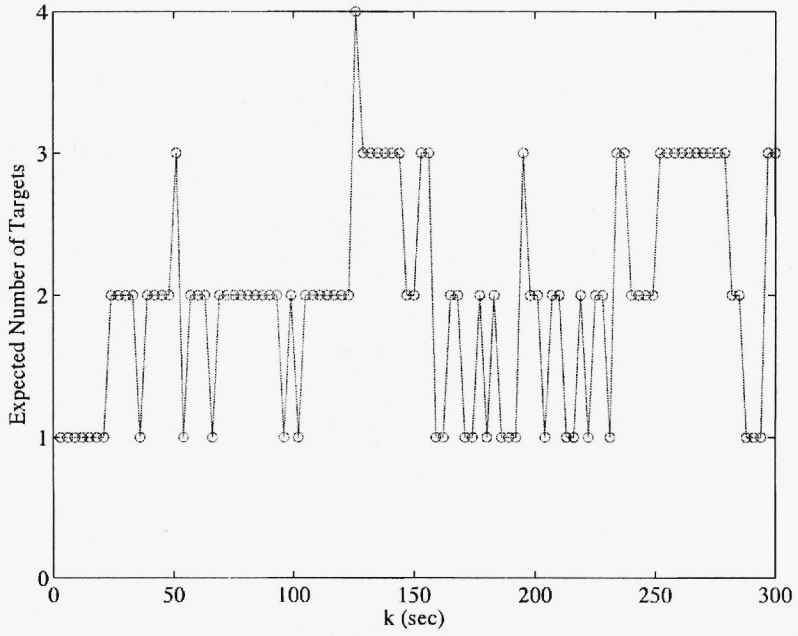


Fig. 6. Expected number of targets at each time step in the two-target simulation with multipath effects.

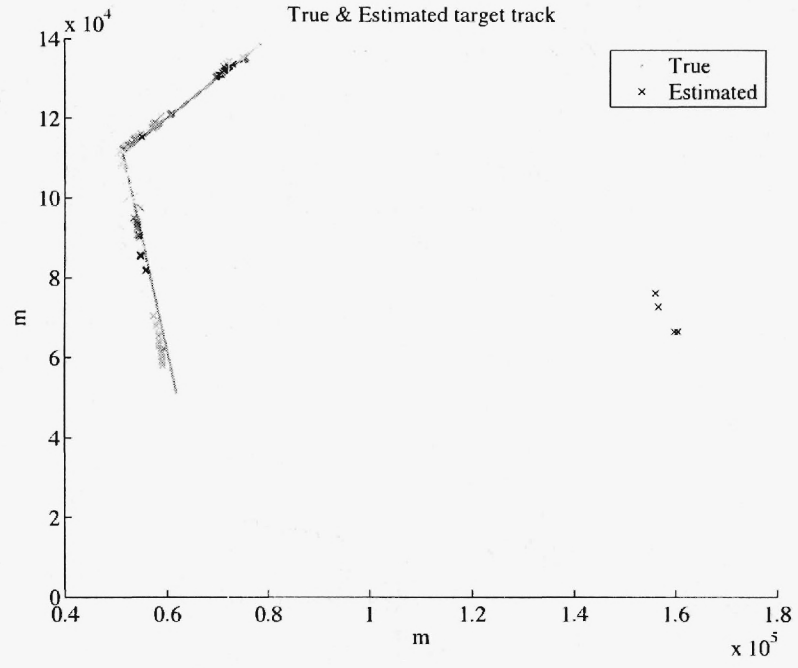


Fig. 7. Actual vs. estimated target locations over the first 300 iterations of the two-target simulation with multipath effects.

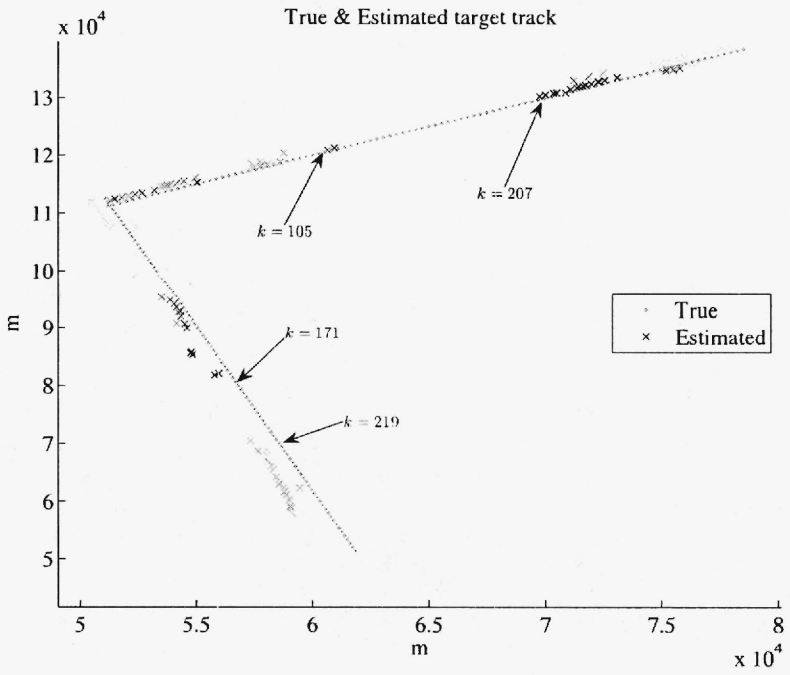
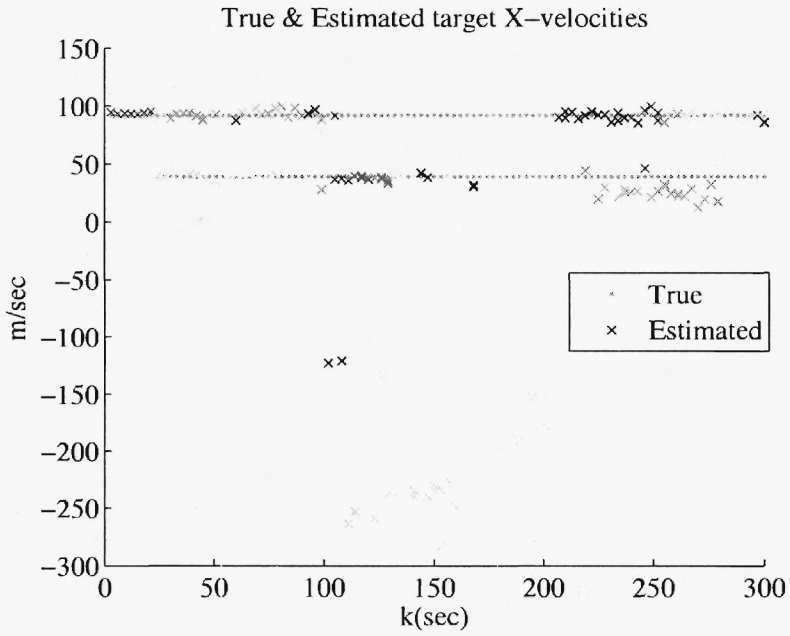
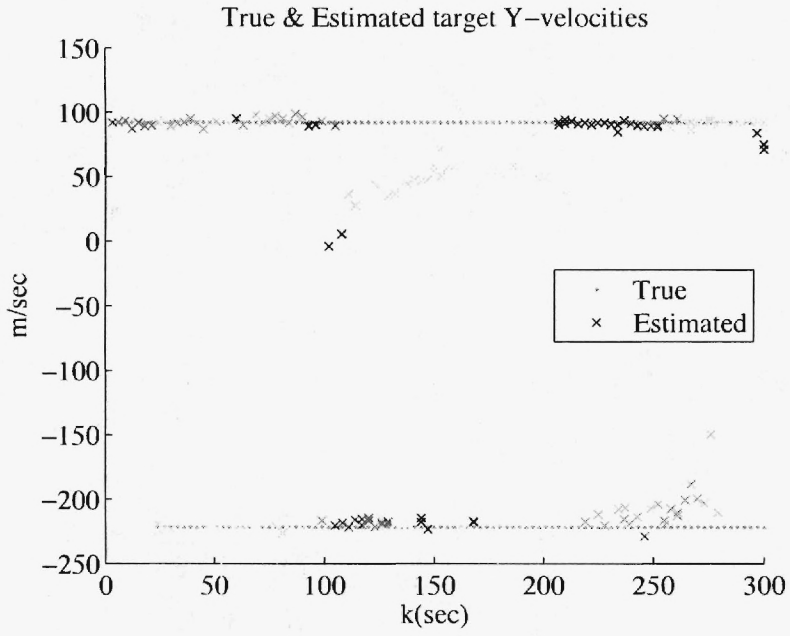


Fig. 8. Close-up of actual vs. estimated target locations over the first 300 iterations of the two-target simulation with multipath effects. The time step (k) in the simulation is indicated for a few points of interest.



(a) $\dot{x}$



(b) $\dot{y}$

Fig. 9. Actual vs. estimated target velocities over the first 300 iterations of the two-target simulation with multipath effects.

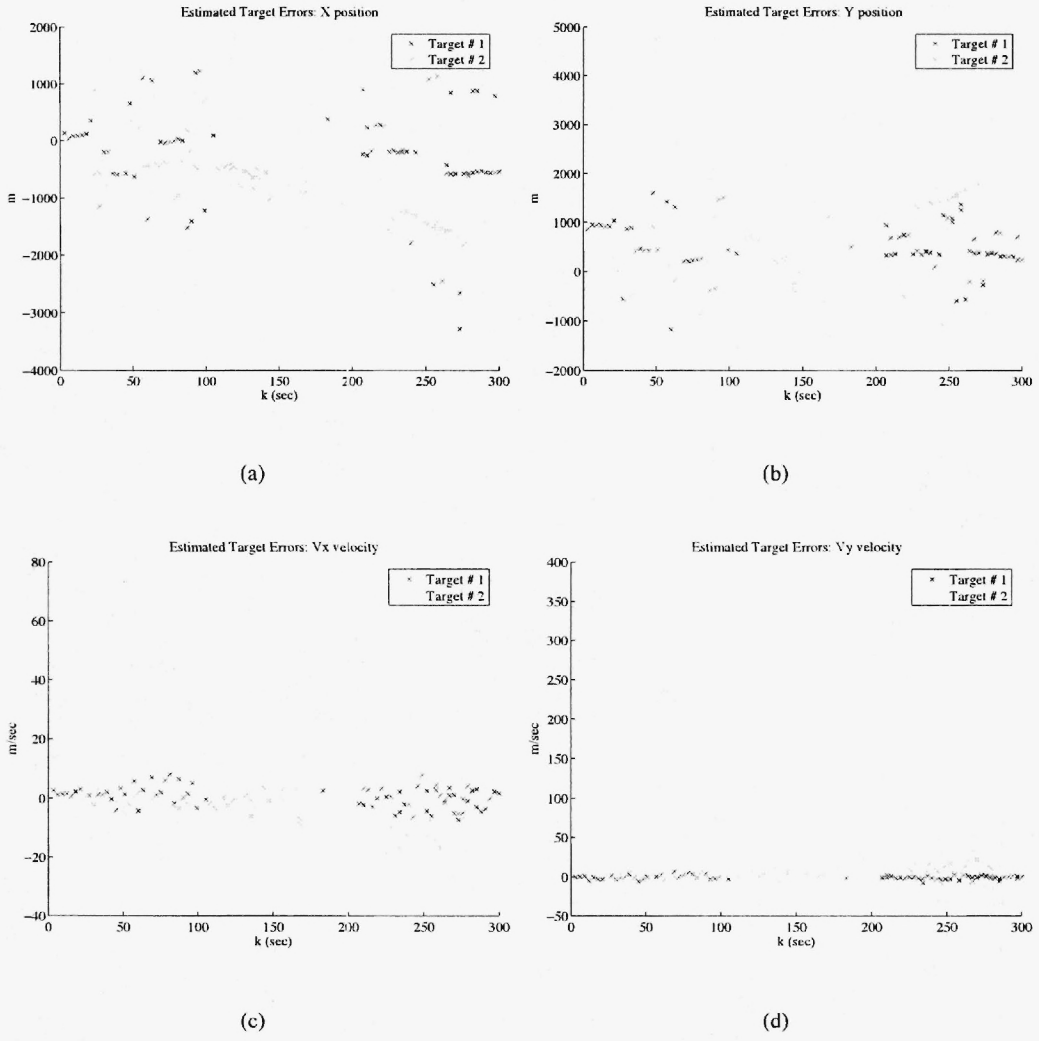


Fig. 10. Errors in target state estimation by the PHD filter in the two-target simulation with multipath effects. The range resolution is around 6.7 km, and the Doppler resolution is a little over 3 m/sec.

B. Additional Experimental Scenarios

Additional simulation experiments were performed to better gauge the effect of multipath on the PHD filter. Brief descriptions of each experiment and the corresponding results are given below.

a) Experiment 1: To obtain a baseline for the performance of the PHD filter, the true p_D and SNR values were used in the Data Update step of the PHD filter. This assumes that the PHD filter knows the true altitude of the target. The result was similar to that of the initial multipath simulation of Section II-A, except that Target 1 was almost always detected. Only Target 2 was dropped in the area of low SNR. This experiment also manifested more overestimation of target number than in the original multipath simulation, in which the PHD filter did not know the true p_D and SNR values.

b) Experiment 2: Instead of using the true p_D and SNR in the Data Update step of the PHD filter, the lowest possible p_D values were used by assuming the worst-case multipath power ratio, i.e., $\min(F_R^2 F_T^2)$, for the whole field of view (FoV). The result was that the number of targets was extremely overestimated by the PHD filter. One possible reason for this is that the range likelihood used by the filter was too broad. Since the assumed SNR was too low, the range variance used to compute the likelihood was too large. Thus, the filter considered too many ellipses to be intersections (and thus, possible targets), even if they were separated by large distances.

This simulation was also run with false-alarm suppression to determine whether the extreme target number overestimation would still occur if no false alarms were present. The filter's false alarm parameters were kept as is, so the PHD filter still expected false alarms; however, no false alarms were allowed to occur in the simulated data. This would lead us to expect a slight underestimation by the PHD filter of expected target number. Nevertheless, overestimation still occurred in the absence of false alarms.

Thus, where the p_D used by the PHD filter is smaller than the real p_D in the FoV, the overestimation may be the same problem seen when birth particles were spontaneously placed in areas of low p_D, as was surmised in [4]. Because the PHD filter assumes that the p_D is low, it believes that there are targets present that it cannot observe. Yet, because the actual p_D is higher, the PHD filter does receive observations from the targets. This appears to have a feedback effect in the PHD filter and causes considerable overestimation of the number of targets present.

c) Experiment 3: The largest possible p_D values were now used in the Data Update step of the PHD filter by assuming the largest multipath power ratio, i.e., $\max(F_R^2 F_T^2)$, for the entire FoV. The result was that the number of targets was rarely overestimated, but the PHD filter had a hard time detecting both targets. This was expected, given the results of Experiment 2, since in this case, the PHD filter assumed an SNR that was too high, and the range variance used to compute the likelihood was too small. Thus, the filter most likely missed valid observation intersections.

d) Experiment 4: In this simulation, as in Section II-A, it was assumed that the PHD filter did not know the true p_D and SNR values to use in the Data Update step. However, the offline range-variance based grid used to find the ellipse intersections was recomputed assuming that it knew the true target altitude and the true multipath-dependent SNR. The result was that both targets were lost and a ghost target was tracked where the p_D drops significantly in one or more of the transmitters used. The PHD filter again occasionally overestimated the number of targets present. This poor performance is expected, since even though the birth particles are being placed according to the multipath truth, the sensor likelihoods used in the PHD filter to compute the particle weights do not correctly model the multipath.

e) Experiment 5: Both the ellipse-intersection finding grid and the Data Update step were assumed to know the correct p_D and SNR to use in this simulation. The result was that Target 1 was detected most of the time, Target 2 was undetected while in the region of low SNR, and the overestimation of the number of targets was as severe as in experiment 1 when just the PHD filter knew the multipath truth.

f) Experiment 6: The original multipath simulation, in which neither the PHD filter, nor the range-variance based grid, know the true SNR or p_D caused by the multipath, was re-run, including the same false alarm parameters; however, in this new experiment, false alarms were prevented from occurring during the simulation run. Just as in the original multipath simulation (see Figures 6-10), the PHD filter fails to detect the two targets while they travel in areas of low SNR. However, this Experiment 6 exhibits almost no overestimation of the expected number of targets, and considerably fewer ghost targets are detected than in the original multipath simulation. However, since the false alarm parameters specified do indicate the presence of false alarms, the PHD filter expects false alarms, and we expect it to underestimate slightly the number of targets.

g) Experiment 7: The previous experiment with suppressed false alarms was repeated, but the PHD filter and range-variance based grid were given knowledge of the true p_D and

SNR caused by the multipath, as in Experiment 5. As in Experiment 5, Target 2 was not detected in the region of low SNR. Target 1 was detected almost always, which is slightly more than in Experiment 5. The main differences between this simulation and Experiment 5 are the almost complete absence of ghost targets and much better target number estimation performance.

III. Conclusions

The simulation experiments in the previous section give us some insight into the robustness of the PHD filter. When the actual probabilities of detection and signal to noise ratios differ from those assumed by the PHD filter, target tracking performance suffers. We find that it is not suffering from birth particle placement issues, since adding multipath truth information into the range-variance based grid did not affect the simulation results. It is the incorrectness of the sensor likelihood functions, caused by the discrepancy between the true p_D and SNR and the p_D and SNR assumed by the PHD filter, that affects the tracking performance. This is evident multiple times in the experiments of the previous section.

In the case where the PHD filter overestimates the SNR, as in Experiment 3, the sensor likelihood function is too tight, and the PHD filter has a difficult time detecting the targets. In the case where the PHD underestimates the SNR, as in Experiment 2, the sensor likelihood function is too loose, and the PHD filter overestimates the number of targets present, even in the absence of false alarms. Another reason for the overestimation is that the PHD does not expect to see the targets given the underestimated SNR. Yet, because the SNR is really high, it still receives target observations and thus overestimates the number of targets present. When the true values of p_D and SNR were given to the PHD filter, as in Experiments 1 and 5, the tracking performance improved. Only one of the targets remained undetected when traveling through an area of low SNR.

Experiments 6 and 7 attempted to probe further the issue of the target number overestimation by the PHD filter, which was present in these multipath simulations even when the PHD filter knew the true p_D and SNR values. When false alarms were suppressed in the simulation, the overestimation problem, as well as the detection of ghost targets, almost vanished. This leads us to conjecture that the overestimation problem is different from that in [4], which was attributed to the $(1 - p_D)$ term in the Data Update equation of the PHD filter when spontaneously placing birth particles in areas of low p_D. The current overestimation problem, when the true p_D and SNR are known, appears to be due to an interplay between

false alarms and the ambiguity in triangulating targets for which not all sensors have high p_D. Thus, in the presence of false alarms, the PHD filter is highly sensitive to the probabilities of detection and signal-to-noise ratios present in the tracking scenario.

One promising direction for future work is the *cardinalized* PHD approach recently proposed by Mahler [10], [11] (as well as an associated approximation known as the "binomial filter" [12]), which purports to be more robust than the original PHD approach in terms of maintaining correct estimates of the number of targets. Post-processing of the PHD sequence, as explored by Lin, Bar-Shalom, and Kirubarajan [13], provides another avenue for investigation. Our results on using the PHD filter for tracking in multipath environments provide further evidence for the need to explore new approaches such as these.

ACKNOWLEDGMENTS

This work was enabled via startup funds from the School of Electrical Engineering at the Georgia Institute of Technology, the Demetrius T. Paris Junior Professorship, U.S. Air Force Office of Scientific Research grant F49620-03-1-0340 and the support of Dr. Paul Howland and Dr. Rene van der Heiden of NATO C3 Agency.

REFERENCES

[1] R. P. S. Mahler, "Multitarget Bayes filtering via first-order multitarget moments," *IEEE Trans. Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1152–1178, 2003.

[2] B.-N. Vo, S. Singh, and A. Doucet, "Sequential Monte Carlo methods for multi-target filtering with random finite sets," *IEEE Trans. Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1224–1245, October 2005.

[3] D. Clark and J. Bell, "Bayesian multiple target tracking in forward scan sonar images using the PHD filter," *IEE Proc. Radar, Sonar and Navigation*, vol. 152, no. 5, pp. 327–334, October 2005.

[4] M. Tobias and A. D. Lanterman, "Probability hypothesis density-based multitarget tracking with bistatic range and Doppler observations," *IEE Proc. Radar, Sonar and Navigation*, vol. 152, pp. 195–205, June 2005.

[5] M. Tobias, "Probability hypothesis densities for multitarget, multisensor tracking with application to passive radar," Ph.D. dissertation, Georgia Institute of Technology, Atlanta, Georgia, 2006, available at http://hdl.handle.net/1853/10514.

[6] M. Tobias and A. D. Lanterman, "Techniques for birth particle placement in the PHD particle filter applied to passive radar," *IET Radar, Sonar & Navigation*, submitted April 2007.

[7] P. E. Howland, D. Maksimiuk, and G. Reitsma, "FM radio based bistatic radar," *IEE Proc. Radar, Sonar, and Navigation*, vol. 152, no. 3, pp. 107–115, June 2005.

[8] D. Barton, *Modern Radar System Analysis*. Artech House, Inc., 1988.

[9] O. Erdinc, P. Willett, and Y. Bar-Shalom, "Probability hypothesis density filter for multitarget multisensor tracking," in *Proc. 8th International Conf. on Information Fusion*, Philadelphia, Pennsylvania, July 2005.

[10] R. Mahler, "A theory of PHD filters of higher order in target number," in *Signal Processing, Sensor Fusion, and Target Recognition XV*, I. Kadar, Ed., vol. SPIE Proc. 6235, 2006.

[11] O. Erdinc, P. Willett, and Y. Bar-Shalom, "A physical-space approach for the probability hypothesis density and cardinalized probability hypothesis density filters," in *Signal and Data Processing of Small Targets 2006*, O. Drummond, Ed., vol. SPIE Proc. 6236, 2006.

[12] R. Mahler, "PHD filters of second order in target number," in *Signal and Data Processing of Small Targets 2006*, O. Drummond, Ed., vol. SPIE Proc. 6236, 2006.

[13] L. Lin, Y. Bar-Shalom, and T. Kirubarajan, "Track labeling and PHD filter for multitarget tracking," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 42, no. 3, pp. 778–795, July 2006.

Appendix I

L.M. Ehrman and A.D. Lanterman, "Robust Algorithm for Automated Target Recognition using Precomputed Radar Cross Sections," *Automatic Target Recognition XIV*, Proc. SPIE 5426, Ed: F.A. Sadjadi, April 12-16, 2004, pp. 197–208.

A Robust Algorithm for Automated Target Recognition Using Precomputed Radar Cross Sections

Lisa M. Ehrman and Aaron D. Lanterman

Center for Signal and Image Processing
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332, USA

ABSTRACT

Passive radar is an emerging technology that offers a number of unique benefits, including covert operation. Many such systems are already capable of detecting and tracking aircraft. The goal of this work is to develop a robust algorithm for adding automated target recognition (ATR) capabilities to existing passive radar systems.

In previous papers,[1,2] we proposed conducting ATR by comparing the precomputed RCS of known targets to that of detected targets. To make the precomputed RCS as accurate as possible, a coordinated flight model is used to estimate aircraft orientation. Once the aircraft's position and orientation are known, it is possible to determine the incident and observed angles on the aircraft, relative to the transmitter and receiver. This makes it possible to extract the appropriate radar cross section (RCS) from our simulated database. This RCS is then scaled to account for propagation losses and the receiver's antenna gain. A Rician likelihood model compares these expected signals from different targets to the received target profile.

We have previously employed Monte Carlo runs to gauge the probability of error in the ATR algorithm; however, generation of a statistically significant set of Monte Carlo runs is computationally intensive. As an alternative to Monte Carlo runs, we derive the relative entropy (also known as Kullback-Liebler distance) between two Rician distributions. Since the probability of Type II error in our hypothesis testing problem can be expressed as a function of the relative entropy via Stein's Lemma, this provides us with a computationally efficient method for determining an upper bound on our algorithm's performance. It also provides great insight into the types of classification errors we can expect from our algorithm. This paper compares the numerically approximated probability of Type II error with the results obtained from a set of Monte Carlo runs.[3]

Keywords: automatic target recognition, passive radar, coordinated flight model, radar cross section, relative entropy

1. BACKGROUND

Two parallel schools of thought dominate the literature regarding the recognition of fast-moving fixed-wing aircraft. The first proposes a two-step approach to the problem. Target images are created, such as two-dimensional inverse synthetic aperture radar (ISAR) images or a sequence of one-dimensional range profiles.[4] Target recognition is then conducted using these images. The alternate approach has been to bypass the creation of images and attempt recognition directly from the received data. Herman[5,6] takes this second approach to automatic target recognition (ATR), using data obtained from a passive radar system.

Although ATR has been a subject of much research, Herman's application of passive radar was innovative. Unlike traditional radar systems, passive radar systems bypass the need for dedicated transmitters by exploiting "illuminators of opportunity" such as commercial television and FM radio signals. In doing so, they are able to reap a number of benefits. Most notably, the fact that passive radar systems do not emit energy renders them covert. An additional benefit is that the illuminators of opportunity often operate at much lower frequencies than their traditional counterparts. These low-frequency signals are well-suited for ATR.[7–9] Several passive radar systems have been developed in recent years, with Lockheed Martins' Silent Sentry and John Sahr's Manastash Ridge Radar[10,11] being two well-known examples.

(Tel: 404-385-2548. Fax: 404-894-8363. Email: ehrman@ece.gatech.edu, lanterma@ece.gatech.edu)

2. OUR APPROACH

Our approach to the problem falls under the second school of thought regarding ATR. We intend to identify aircraft models using the Radar Cross Section (RCS) obtained from a passive radar system as our key parameter for classification. This is accomplished by comparing the RCS of the true target to the precomputed RCS of known targets in a target library. Since the RCS is the sole parameter for target identification in this scheme, its accurate representation is paramount.

We divide the process of simulating the RCS of known targets into four basic steps. First, we assume that the passive radar system can accurately track the target and provide us with its location. Using the time-correlated aircraft positions, we estimate the aircraft orientation via a coordinated flight model.[12] Given that the aircraft position and orientation are known, we can then compute the incident and observed azimuths and elevations on the aircraft, relative to the transmitter and receiver. Second, using the incident and observed angles, we access an RCS database* containing information for each aircraft in the target class. The appropriate RCS data are extracted from the database, yielding time-correlated RCS profiles for each aircraft in the target class, as though each aircraft is executing the same maneuver as the true target. Next, the RCS profiles are scaled to account for propagation losses, using the Advanced Refractive Effects Prediction System (AREPS). Finally, the profiles are scaled using the Numerical Electromagnetic Code (NEC2) to account for the antenna gain of the receiver. This process results in a set of power profiles simulating those that should arrive at the receiver if each aircraft in the target library were to execute the same maneuver as the true target.

The passive radar system being modeled as part of this research is actually under development by NATO/NC3A. We are proceeding with the intent of eventually comparing our precomputed power profiles to the profiles of targets actually detected by the NATO/NC3A system. Until then, we must simulate the profiles of the true targets. This process is similar to the one just described, with two notable differences. First, when simulating the received data, we use the real aircraft orientation angles rather than those estimated by the coordinated flight model. The second major difference is that we assume the received target profiles are corrupted by noise. The effect of noise on the received power profile is modeled by,

$$P_{RECEIVED} = (\sqrt{P} + w_R)^2 + w_I^2, \tag{1}$$

where P is the power profile prior to being corrupted by noise, and w is zero-mean additive white Gaussian noise, which has real and imaginary components, w_R and w_I.[6] This yields a Rician likelihood model for comparing the received power profile to the precomputed power profiles of targets in the target library. The aircraft type giving the largest likelihood is declared to be the target type.

2.1. Relative Entropy Between Two Ricians

Since the classification scheme is essentially just an M-ary hypothesis testing problem, it seems intuitive that the algorithm's ability to correctly classify the targets is largely determined by the amount of overlap between the target distributions. If the probability density functions of the targets in the target library overlap quite a bit, then the ATR algorithm is expected to have difficulty discriminating between the aircraft; conversely, when the target densities are widely separated, the ATR algorithm should be able to classify the targets with a high rate of success.

Although the ATR algorithm uses an M-ary hypothesis testing scheme, a great deal of insight into the types of classification errors likely to be made by the algorithm is derived by considering several binary discrimination tests. By determining how widely separated any two target densities are, we can estimate the likelihood that one aircraft will be misidentified as another. The relative entropy quantifies this distance between any two target density functions, $p(x)$ and $q(x)$.[13] Since the data points are independent (conditioned on a particular hypothesis) in our case, we can write the relative entropy as the sum of the relative entropy for the original data points using,

*The RCS database was created using the Fast Illinois Solver Code, FISC.

$$D(p(x)||q(x)) = \sum_{i=1}^{N} \left\{ \int_{-\infty}^{\infty} p_i(x_i) \ln \left\{ \frac{p_i(x_i)}{q_i(x_i)} \right\} dx_i \right\}. \tag{2}$$

Given our noise model, it follows that $p(x)$ and $q(x)$ are Rician densities. Thus, they are expressed as

$$p_i(x_i) = \frac{x_i}{\sigma^2} e^{\frac{-(x_i^2 + s_{1i}^2)}{2\sigma^2}} I_0 \left\{ \frac{x_i s_{1i}}{\sigma^2} \right\}, \tag{3}$$

and

$$q_i(x_i) = \frac{x_i}{\sigma^2} e^{\frac{-(x_i^2 + s_{2i}^2)}{2\sigma^2}} I_0 \left\{ \frac{x_i s_{2i}}{\sigma^2} \right\}. \tag{4}$$

Substituting (3) and (4) into (2) reveals that the relative entropy between two Rician densities with the same σ^2 is

$$D(p(x)||q(x)) = \sum_{i=1}^{N} \left\{ \int_{0}^{\infty} \frac{x_i}{\sigma^2} e^{\frac{-(x_i^2 + s_{1i}^2)}{2\sigma^2}} I_0 \left\{ \frac{x_i s_{1i}}{\sigma^2} \right\} \left\{ \frac{s_{2i}^2 - s_{1i}^2}{2\sigma^2} + \ln \left\{ I_0 \left\{ \frac{x_i s_1}{\sigma^2} \right\} \right\} - \ln \left\{ I_0 \left\{ \frac{x_i s_2}{\sigma^2} \right\} \right\} \right\} dx_i \right\}. \tag{5}$$

Two types of errors are possible in a binary hypothesis testing problem. For ease of notation, let us temporarily denote aircraft #1 as the aircraft whose probability density function is $p(x)$; similarly, let us temporarily define aircraft #2 as the target whose probability density function is $q(x)$. The event in which aircraft #1 is misidentified as aircraft #2 is called a Type I error. The probability of such an error is denoted by α. A Type II error occurs if aircraft #2 is mistakenly identified as aircraft #1. Denote the probability of a Type II error as β.

In a Neyman-Pearson scheme, α is set to be some arbitrarily small number, and the decision threshold separating the acceptance regions for aircrafts #1 and #2 shifts to keep α constant as N increases. Within this context, we can use Stein's Lemma[13, 14] to approximate β as

$$\beta_{p(x)||q(x)} \approx e^{-D(p(x)||q(x))}. \tag{6}$$

The subscript on β has been added to clarify which two distributions are being compared. To avoid confusion, the order of $p(x)$ and $q(x)$ in the subscript is kept consistent with the order in the definition of $D(p(x)||q(x))$. Using this notation, $\beta_{p(x)||q(x)}$ is the probability that aircraft #2 (whose density is $q(x)$) will be misidentified as aircraft #1 (whose density is $p(x)$).

In later sections, we compute β for each possible aircraft pairing. Since our target library is currently comprised of four targets, this results in a total of sixteen ordered pairings. The order of the pairing does matter; it should be clear from examining (5) that the relative entropy for this problem is not symmetric. Note that in (5), the x_i parameter is the dummy variable of the integral. This implies that we do not actually need to substitute in the power profile of a particular true target; rather, x_i is swept from zero to infinity. As a consequence, there is no need for Monte Carlo runs when computing the relative entropy. This significantly reduces the computational complexity involved in determining algorithm performance.

2.2. Anticipated Noise Levels

The only parameter in (5) yet to be determined is the noise power, σ^2. This is expressed in dBW as,

$$P_N = \frac{kT_0N_F}{CPI}, \tag{7}$$

where k is Boltzmann's constant, T_0 is temperature in Kelvin, N_F is the unitless noise figure, and CPI is the coherent processing interval of the system.[15] To match the NATO system, the CPI is set equal to 0.5 seconds, and T_0 is set equal to 290 K. Determining the appropriate noise figure of the system is more difficult. In an urban environment, a reasonable upper bound on the noise figure due to thermal noise and out-of-band interference might be 30 dB. However, transmitter interference must also be addressed. Typically, the direct path interference manifests itself as a spike in the cross-ambiguity function. Since the transmitter's power and location are known, and since the direct path interference spike occurs along the axis with zero velocity, this spike can usually be identified and removed. The more treacherous effect of the transmitter interference is that it can raise the "thumbtack" noise floor of the ambiguity function, potentially masking a target spike. To be thorough, this should also be considered when computing the noise figure.

If the ambiguity function is normalized such that the direct path spike has unit height, then the average pedestal height, or sideband power, is given by

$$P_{pedestal} = \frac{1}{B \times CPI}, \tag{8}$$

where B is the signal bandwidth, and CPI is the coherent processing interval.[16] To match the NATO/NC3A system, values of 45 kHz and 0.5 seconds are used for B and CPI, respectively.

If propagation losses and antenna gain are neglected, the pedestal power is 44 dBW below the direct-path spike. Since the illuminator of opportunity exploited by the NATO/NC3A system has a power level of 50 dBW, the sideband power is 6 dBW. Propagation losses and antenna gain play a significant role, lowering the pedestal power by 95 dBW. The electronics in the receiver of the NATO/NC3A system also mitigate the problem by suppressing the direct path signal by 70 dBW, which reduces the sideband power to -159 dBW. More sophisticated filters could be implemented to further reduce the noise figure, but using the specifications of the system being modeled, the effective noise figure falls between 40 and 45 dB. Thus, the effects due to transmitter interference are far more significant than those due to thermal noise and out-of-band interference.

This underscores an important point. In the interest of gauging the algorithm's performance in the presence of noise, the results presented in Section 3 correspond to a broad range of noise figures. When reading that section, it is imperative to recall that the anticipated noise figure in the real system is only 40 dB.

3. RESULTS

A series of maneuvers are used to test the ATR algorithm. The first maneuver is simply a straight-and-level flight path in which the aircraft flies directly away from the receiver at a speed of 200 m/s and an altitude of 8000 m AGL (above ground level). The second maneuver is nearly identical to the first, but the aircraft heading is rotated 90° so that the aircraft flies broadside to the receiver. This provides the receiver with a wider array of observed azimuths on the aircraft. The third encounter was recorded on-board a maneuvering F-15†, providing us with a more difficult test of our algorithm. All three maneuvers are conducted approximately 30 km from the receiver, and are located in the receiver's main lobe.

†The F-15C trajectory was obtained, courtesy of Major Larkin Hastriter and Lt. Col. Adam MacDonald, from the Joint Helmet Cuing System, Mission JH-16, conducted by the 445th Flight Test Squadron at Edwards Air Force Base in May 2000.

3.1. Straight-and-Level Trajectory #1

The precomputed power profiles corresponding to the first straight-and-level trajectory are shown in Figure 1a. Recall that this maneuver involves the aircraft flying directly away from the receiver. Since the range of aspects of the aircraft presented to the receiver is very limited, it is not surprising that the power profiles of all four aircraft look very similar. Four hundred Monte Carlo runs are conducted at each noise figure in the study, with a quarter of the runs corresponding to each aircraft being the true target. The noise figure is swept from 30 dB to 100 dB in increments of 5 dB. As mentioned in Section 2.2, the noise figure is swept well beyond the anticipated noise levels so that we can witness the breaking point of the algorithm. The probability of error determined by the Monte Carlo runs is shown in Figure 1b. The similarities among the probability-of-error curves of the four aircraft corroborate the point that the precomputed power profiles are quite similar.

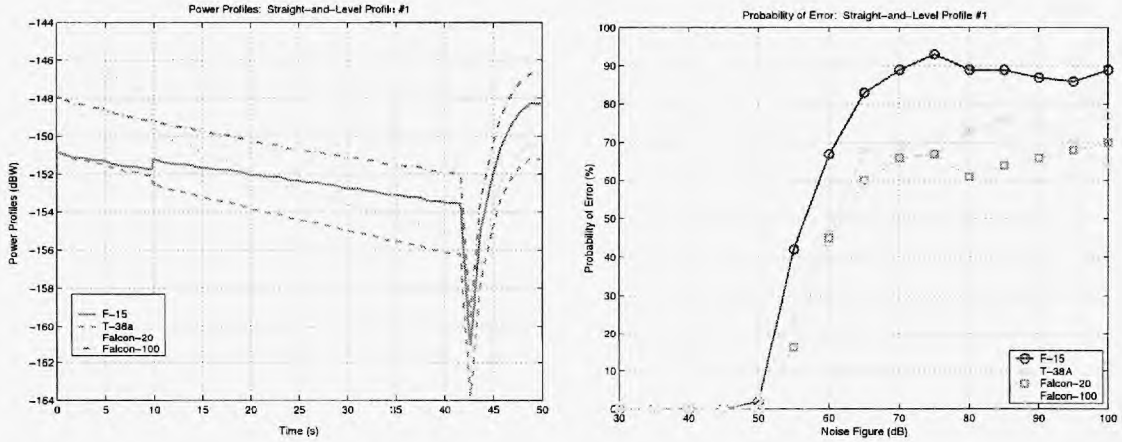


Figure 1. Straight-and-Level Trajectory #1: a.) Power Profiles (left), b.) Probability of Error Vs. Noise Figure (right)

Confusion matrices are tabulated to provide additional insight into the types of identification errors exhibited by the algorithm. The confusion matrices for the first straight-and-level maneuver at noise levels of 55 and 60 dB are shown in Tables 1 and 2. The aircraft listed in the first column correspond to the true target. The aircraft listed across the top row pertain to the aircraft chosen by the algorithm.

Several conclusions may be drawn from the probability of error curves and confusion matrices. For example, the algorithm performs perfectly until the noise figure reaches 50 dB.[‡] By 70 dB, the probability of error, averaged over all four aircraft, approaches 75%. Since there are four aircraft in the study, this is equivalent to chance. The confusion matrices also provide insight into the likeliest misidentifications made by the algorithm. For example, they indicate that if an aircraft is misidentified as the F-15, then the true target is likeliest to have been a T-38A. The probabilities that it was actually a Falcon-100 or Falcon-20 are lower. Similarly, when the algorithm mistakenly chooses an aircraft as the T-38A, it is likeliest that the true target was really the F-15. Under these circumstances, the probability that it was actually a Falcon-100 or Falcon-20 is lower. We can also deduce that the Falcon-100 is most likely to be the true target when the algorithm mistakenly chooses an aircraft as the Falcon-20. The F-15 and T-38A had similar probabilities of being the true target in this setting, but had lower probabilities than the Falcon-100. Finally, when the algorithm mistakenly chooses the Falcon-100, the F-15 is the likeliest to be the true target, followed by the Falcon-20 and T-38A.

Equation 6 is used to approximate β for each possible target pairing; these results, shown in Figure 2, are compared to the results of the Monte Carlo runs. For example, Figure 2a shows the probability that each aircraft will be misidentified as the F-15. Similar findings are given in Figures 2b, 2c, and 2d for the T-38A,

[‡]We would not actually expect the algorithm to perform perfectly in a real setting, as there may be factors that we have neglected to model. However, the simulation results do indicate that the algorithm would perform well at low noise levels.

Table 1. Confusion matrix for straight-and-level trajectory #1 with noise figure = 55 dB. The aircraft listed in the first column correspond to the true target. The aircraft listed across the top row pertain to the aircraft chosen by the algorithm.

Aircraft	F-15	T-38A	Falcon-20	Falcon-100
F-15	58	24	3	15
T-38A	21	75	3	1
Falcon-20	1	5	84	10
Falcon-100	14	3	6	77

Table 2. Confusion matrix for straight-and-level trajectory #1 with noise figure = 60 dB.

Aircraft	F-15	T-38A	Falcon-20	Falcon-100
F-15	33	34	13	20
T-38A	20	54	13	13
Falcon-20	10	19	55	16
Falcon-100	22	13	23	42

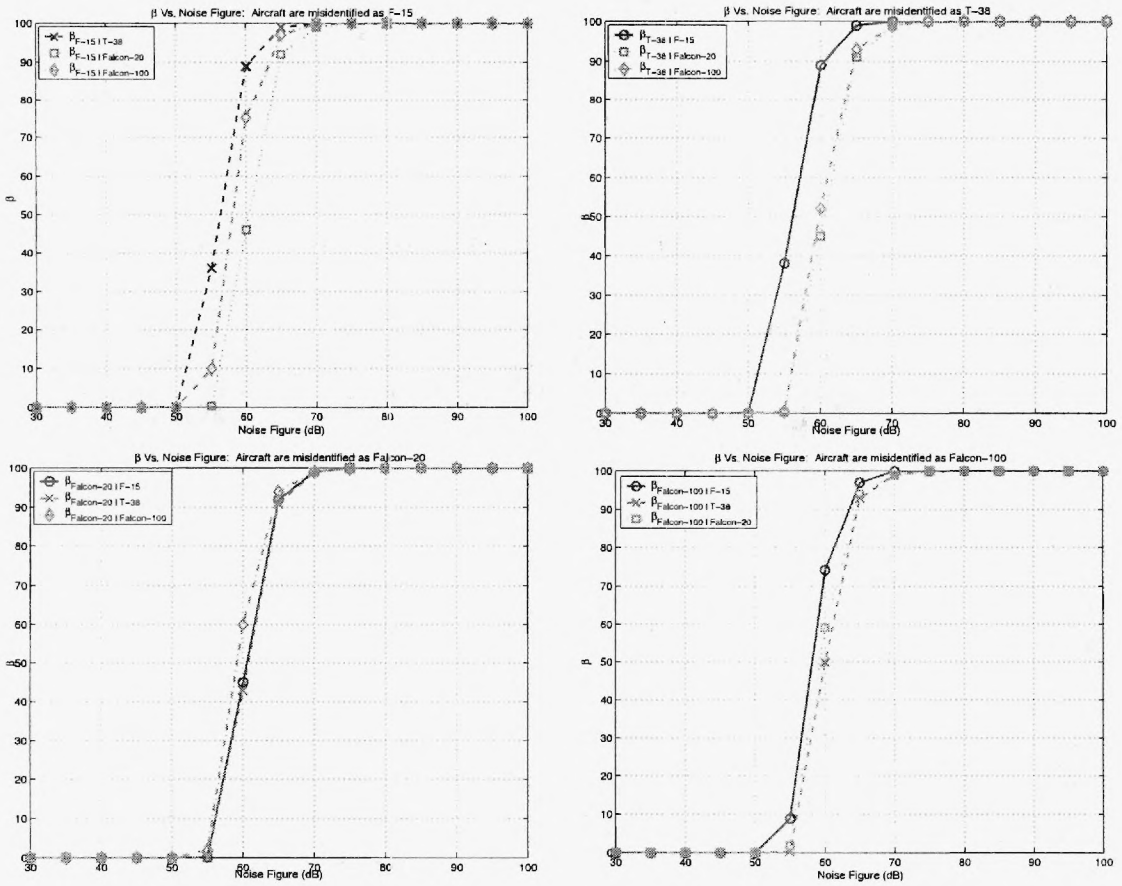


Figure 2. Straight-and-Level Trajectory #1: β Vs. Noise Figure when targets are mistakenly chosen to be the: a.) F-15 (top left), b.) T-38A (top right), c.) Falcon-20 (bottom left), d.) Falcon-100 (bottom right)

Falcon-20, and Falcon-100, respectively. If β is viewed as an upper bound on the algorithm's performance, then it corroborates every one of the trends observed in the Monte Carlo runs. This includes the observation that the probability of error approaches chance when the noise figure reaches 70 dB. This implies that the target distributions overlap almost entirely, which means that β approaches 100% for an infinitesimally small α.

3.2. Straight-and-Level Trajectory #2

The analysis is repeated for the second straight-and-level maneuver. Recall that the aircraft now fly broadside to the receiver. Thus, a much wider range of aspects are presented to the receiver. The manifestation of this is evident in the power profiles, shown in Figure 3a. They look much less similar in shape and amplitude than before. This leads us to anticipate much better performance than was observed in the previous maneuver.

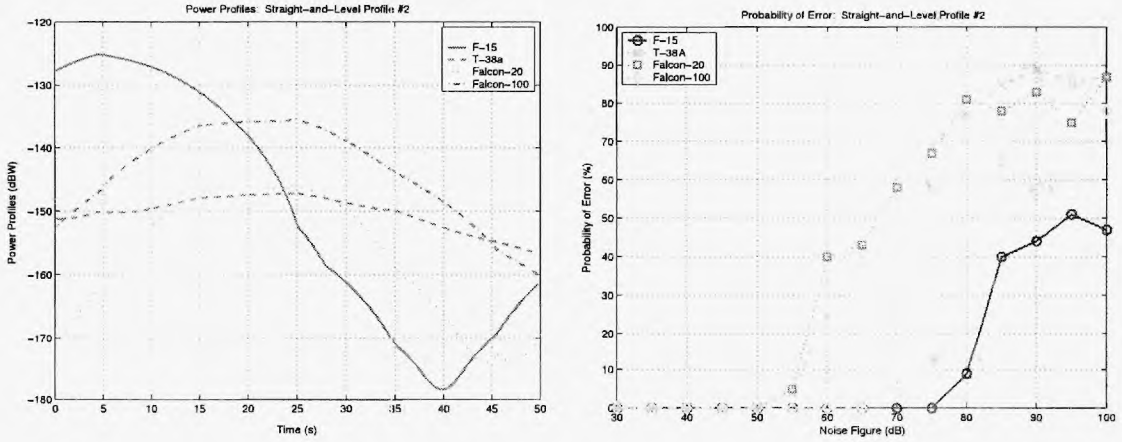


Figure 3. Straight-and-Level Trajectory #2: a.) Power Profiles (left), b.) Probability of Error Vs. Noise Figure (right)

The probability of error curves derived from the Monte Carlo runs, shown in Figure 3b, confirm this suspicion. No errors occur until the noise figure reaches 55 dB, and at this point, they are limited to the T-38A and Falcon-20. The probability of error remains at zero for the Falcon-100 until the noise figure hits 70 dB, and remains at zero for the F-15 until the noise figure reaches 80 dB. Confusion matrices confirming these trends appear in Tables 3, 4, and 5 for noise figures of 55, 70, and 80 dB.

Table 3. Confusion matrix for straight-and-level trajectory #2 with noise figure = 55 dB.

Aircraft	F-15	T-38A	Falcon-20	Falcon-100
F-15	100	0	0	0
T-38A	0	96	4	0
Falcon-20	0	5	95	0
Falcon-100	0	0	0	100

Several conclusions are drawn from this experiment. First, the probability of error curves and confusion matrices suggest that the only errors made for noise figures less than 70 dB pertain to swapping the T-38A and Falcon-20. Not only are the Falcon-100 and F-15 never misidentified when they are the true targets; they are never mistakenly chosen when the true target is the T-38A or Falcon-20. Once the noise figure reaches 70 dB, the Falcon-100 begins being misidentified when it is present, and mistakenly chosen when it is not. Neither of these two events occur with the F-15 until the noise figure reaches 80 dB. The computation of β, presented in Figure 4, is in accord with every one of these conclusions.

Table 4. Confusion matrix for straight-and-level trajectory #2 with noise figure = 70 dB.

Aircraft	F-15	T-38A	Falcon-20	Falcon-100
F-15	100	0	0	0
T-38A	0	52	40	8
Falcon-20	0	50	42	8
Falcon-100	0	2	9	89

Table 5. Confusion matrix for straight-and-level trajectory #2 with noise figure = 80 dB.

Aircraft	F-15	T-38A	Falcon-20	Falcon-100
F-15	91	3	0	6
T-38A	16	23	19	42
Falcon-20	15	24	19	42
Falcon-100	17	17	14	52

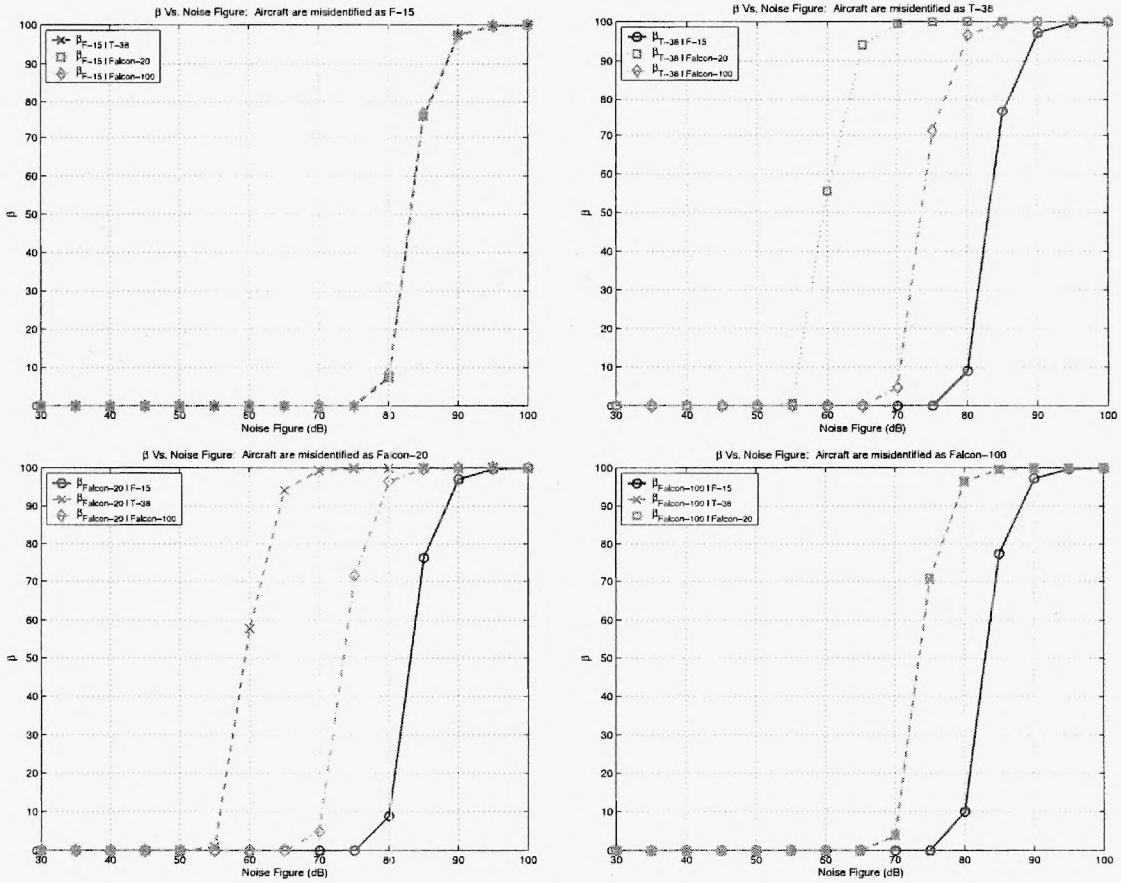


Figure 4. Straight-and-Level Trajectory #2: β Vs. Noise Figure when targets are mistakenly chosen to be the: a.) F-15 (top left), b.) T-38A (top right), c.) Falcon-20 (bottom left), d.) Falcon-100 (bottom right)

3.3. Edwards Maneuver

The final test of the algorithm uses a much more difficult maneuver than the previous two tests. This maneuver is shown in both a 3-D perspective and a God's eye view in Figure 5. Note that this test involves changes in aircraft heading, pitch, and roll.

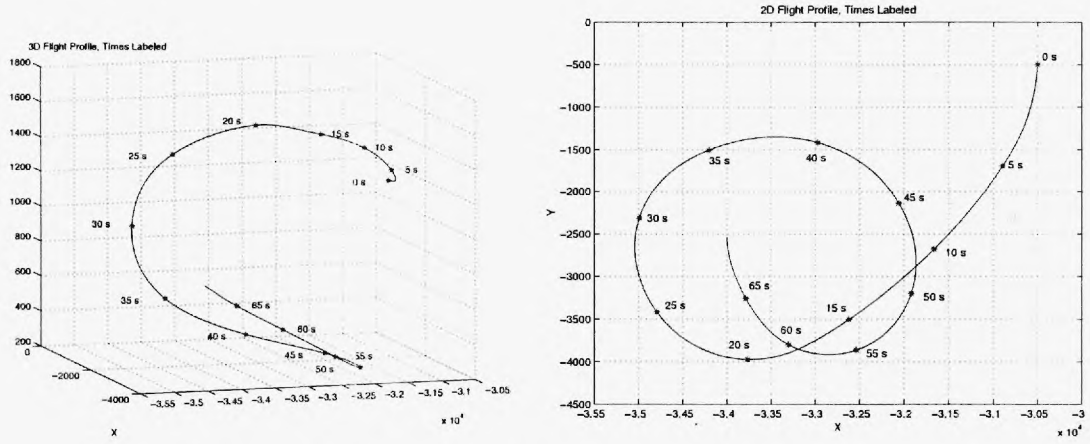


Figure 5. F-15 maneuver: a) 3-D view (left), b) top view (right).

The precomputed power profiles for this maneuver are shown in Figure 6a. The corresponding Monte Carlo probabilities of error are shown in Figure 6b. This time, the shapes of the power profiles vary, but the amplitude of the profiles is fairly similar for all four aircraft models. Confusion matrices are given in Tables 6 and 7 for noise figures of 60 and 70 dB.

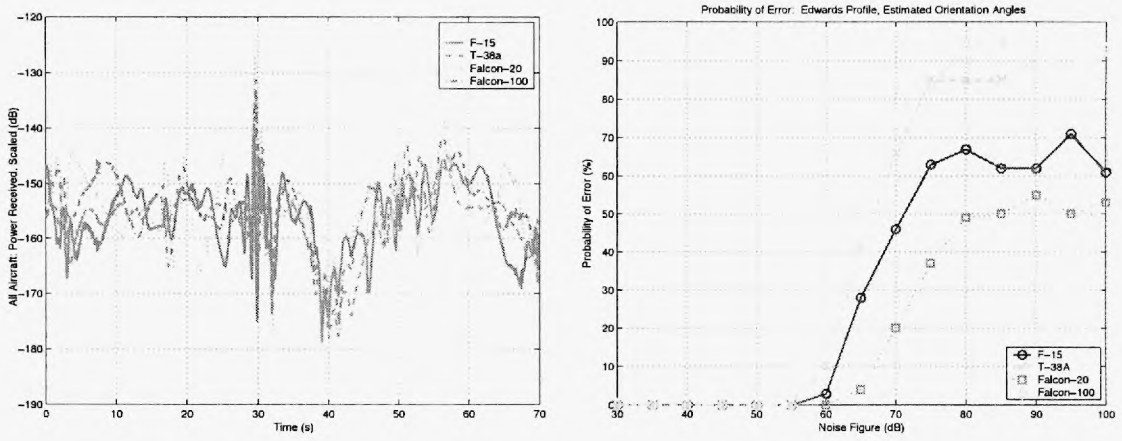


Figure 6. Edwards Trajectory: a.) Power Profiles (left), b.) Probability of Error Vs. Noise Figure (right)

Once again, the probability of error curves and confusion matrices provide us with ample data for making conclusions. This time, no identification errors are recorded until the noise figure reaches 60 dB. The algorithm fails to be effective once the noise figure reaches 80 dB. If a true target is misidentified as the F-15, it was likeliest to have been the T-38A or Falcon-100, with the probability that it was the Falcon-20 being somewhat smaller. Similarly, if the T-38A is mistakenly identified as present when it is not, the true target was most likely the Falcon-100. The Falcon-20 was the least likely aircraft to be present when the algorithm mistakenly identified

Table 6. Confusion matrix for the Edwards trajectory with noise figure = 60 dB

Aircraft	F-15	T-38A	Falcon-20	Falcon-100
F-15	97	1	0	2
T-38A	2	90	0	8
Falcon-20	0	0	100	0
Falcon-100	1	17	0	82

Table 7. Confusion matrix for the Edwards trajectory with noise figure = 70 dB

Aircraft	F-15	T-38A	Falcon-20	Falcon-100
F-15	54	16	16	14
T-38A	27	34	22	17
Falcon-20	10	7	80	3
Falcon-100	30	24	21	25

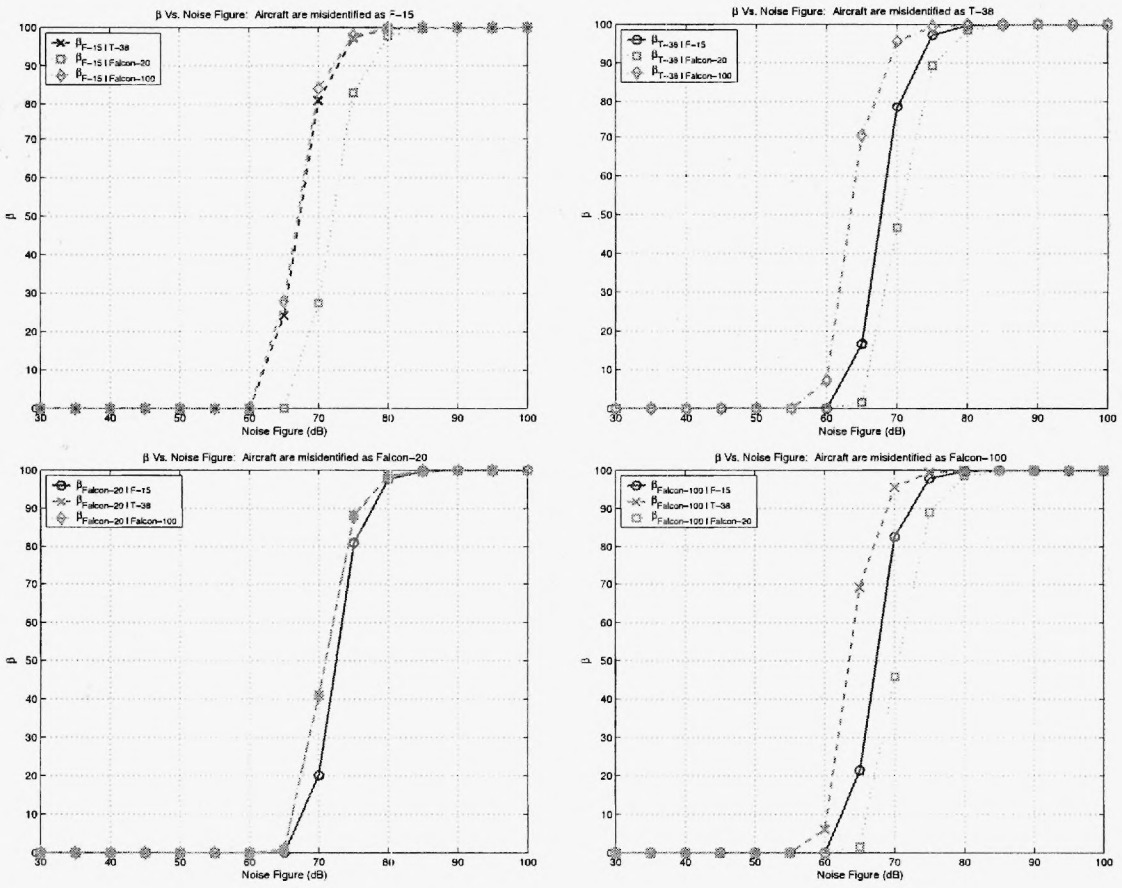


Figure 7. Edwards Trajectory: β Vs. Noise Figure when targets are mistakenly chosen to be the: a.) F-15 (top left), b.) T-38A (top right), c.) Falcon-20 (bottom left), d.) Falcon-100 (bottom right)

the true target as the T-38A. A similar trend is noticeable when the true target is mistakenly identified as the Falcon-100. This time, the T-38A was the most likely aircraft to be the true target, followed by the F-15. Finally, when the Falcon-20 is falsely matched with the true target, the Falcon-100 and T-38A were equally likely to be the true target. The probability that the true target was an F-15 is slightly lower. As before, the plots of β, shown in Figure 7, are in accord with every one of these conclusions.

4. CONCLUSIONS

Two major conclusions can be drawn from this work. First, the simulations suggest that the algorithm will perform very well at the anticipated noise levels. In fact, every one of the simulations resulted in perfect identification of all four aircraft for noise figures below 50 dB. While we do not want to suggest that a real system would function perfectly, as there may be phenomena present in a real system that have not been modeled, the findings do suggest that the real system should perform well at the anticipated noise levels. The data presented in this paper also suggest that β, computed in term of the relative entropy, provides a way to gauge system performance that is both reliable and computationally efficient. Further work will seek to extend this idea to a Bayesian framework by computing Chernoff information.

ACKNOWLEDGMENTS

This work was sponsored by the NATO Consultation, Command, and Control Agency (NC3A) and Air Force Office of Scientific Research (AFOSR) grant F49620-03-1-0340. The authors would like to thank Dr. Paul Howland and Dr. Rene van der Heiden at NC3A for their support. We are also indebted to Major Larkin Hastriter and Lt. Col. Adam MacDonald for their assistance in obtaining aircraft flight paths.

REFERENCES

1. L. Ehrman and A. Lanterman, "Automated target recogntion using passive radar and coordinated flight models," in *Automatic Target Recognition XIII*, **SPIE Proc. 5094**, (Orlando, FL), April 2003.
2. L. Ehrman, *Automatic Target Recognition Using Passive Radar and a Coordinated Flight Model*, Master's Thesis, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 2003.
3. L. Ehrman and A. Lanterman, "Target identification using modeled radar cross sections and a coordinated flight model," in *Proceedings from the Third Multi-National Conference on Passive and Covert Radar*, (Seattle, WA), October 2003.
4. S. Jacobs and J. O'Sullivan, "Automatic target recognition using sequences of high resolution radar range-profiles," *IEEE Trans. on Aerospace and Electronic Systems* **36**(2), pp. 364–382, 2000.
5. S. Herman, *A Particle Filtering Approach to Joint Passive Radar Tracking and Target Classification*, Doctoral Dissertation, Department of Electrical and Computer Engineering, Univ. of Illinois at Urbana-Champaign, Urbana, IL, 2002.
6. S. Herman and P. Moulin, "A particle filtering approach to joint radar tracking and automatic target recognition," in *Proc. IEEE Aerospace Conference*, (Big Sky, Montana), March 10-15 2002.
7. Y. Lin and A. Ksienski, "Identification of complex geomtrical shapes by means of low-frequency radar returns," *The Radio and Electronic Engineer* **46**, pp. 472–486, Oct. 1976.
8. H. Lin and A. Ksienski, "Optimum frequencies for aircraft classification," *IEEE Trans. on Aerospace and Electronic Systems* **17**, pp. 656–665, Sept. 1981.
9. J. Chen and E. Walton, "Comparison of two target classification techniques," *IEEE Trans. on Aerospace and Electronic Systems* **22**, pp. 15–21, Jan. 1986.
10. J. Sahr and F. Lind, "The Manastash ridge radar: A passive bistatic radar for upper atmospheric radio science," *Radio Science* , pp. 2345–2358, Nov.-Dec. 1997.
11. J. Sahr and F. Lind, "Passive radio remote sensing of the atmosphere using transmitters of opportunity," *Radio Science* , pp. 4–7, March 1998.
12. L. Ehrman and A. Lanterman, "Estimation of aircraft orientation from flight paths using a coordinated flight model," *submitted to IEEE Transactions on Aerospace and Electronic Systems* , November 2002.
13. T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc., 1991.

14. A. D. Lanterman, J. A. O'Sullivan, and M. I. Miller, "Kullback-Liebler distances for quantifying clutter and models," *Optical Engineering* **38**, pp. 2134–2146, December 1999.
15. D. Barton, *Modern Radar System Analysis*, Artech House, 1988.
16. M. A. Ringer, G. J. Frazer, and S. J. Anderson, "Waveform analysis of transmitters of opportunity for passive radar," *Surveillance Systems Division, Electronics and Surveillance Research Laboratory* .